UCSD Pascal System Source Listings, Pascal Software

Table of Contents

Version II.0
Scanned May, 2006

```
  5    1     1:0     1     (*                                                    *)
  6    1     1:0     1     (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.   *)
  7    1     1:0     1     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-    *)
  8    1     1:0     1     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE   *)
  9    1     1:0     1     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
 10    1     1:0     1     (*                                                    *)
 11    1     1:0     1     (******************************************************)
 12    1     1:0     1
 13    0     1:0     1 PROGRAM PASCALSYSTEM;
 14    0     1:0     1
 15    0     1:0     1 (*********************************************)
 16    0     1:0     1 (*                                           *)
 17    0     1:0     1 (*    UCSD PASCAL OPERATING SYSTEM           *)
 18    0     1:0     1 (*                                           *)
 19    0     1:0     1 (*    RELEASE LEVEL:  I.3   AUGUST, 1977     *)
 20    0     1:0     1 (*                    I.4   JANUARY, 1978    *)
 21    0     1:0     1 (*                    I.5   SEPTEMBER, 1978  *)
 22    0     1:0     1 (*                    II.0  FEBRUARY, 1978 BD*)
 23    0     1:0     1 (*                                           *)
 24    0     1:0     1 (*    WRITTEN BY ROGER T. SUMNER             *)
 25    0     1:0     1 (*    WINTER 1977                            *)
 26    0     1:0     1 (*                                           *)
 27    0     1:0     1 (*    INSTITUTE FOR INFORMATION SYSTEMS      *)
 28    0     1:0     1 (*    UC SAN DIEGO, LA JOLLA, CA             *)
 29    0     1:0     1 (*                                           *)
 30    0     1:0     1 (*    KENNETH L. BOWLES, DIRECTOR            *)
 31    0     1:0     1 (*                                           *)
 32    0     1:0     1 (*********************************************)
 33    0     1:0     1
 34    0     1:0     1 CONST
 35    0     1:0     1       MMAXINT = 32767;    (*MAXIMUM INTEGER VALUE*)
 36    0     1:0     1       MAXUNIT = 12;       (*MAXIMUM PHYSICAL UNIT # FOR UREAD*)
 37    0     1:0     1       MAXDIR = 77;        (*MAX NUMBER OF ENTRIES IN A DIRECTORY*)
 38    0     1:0     1       VIDLENG = 7;        (*NUMBER OF CHARS IN A VOLUME ID*)
 39    0     1:0     1       TIDLENG = 15;       (*NUMBER OF CHARS IN TITLE ID*)
 40    0     1:0     1       MAXSEG = 15;        (*MAX CODE SEGMENT NUMBER*)
 41    0     1:0     1       FBLKSIZE = 512;     (*STANDARD DISK BLOCK LENGTH*)
 42    0     1:0     1       DIRBLK = 2;         (*DISK ADDR OF DIRECTORY*)
 43    0     1:0     1       AGELIMIT = 300;     (*MAX AGE FOR GDIRP...IN TICKS*)
 44    0     1:0     1       EOL = 13;           (*END-OF-LINE...ASCII CR*)
 45    0     1:0     1       DLE = 16;           (*BLANK COMPRESSION CODE*)
```

```
46   0    1:0     1      NAME_LEN = 23;      [LENGTH OF CONCAT(VIDLENG,':',TIDLENG)]
47   0    1:0     1      FILL_LEN = 11;      [MAXIMUM # OF NULLS IN FILLER]
48   0    1:0     1
49   0    1:0     1  TYPE
50   0    1:0     1
51   0    1:0     1      IORSLTWD = (INOERROR,IBADBLOCK,IBADUNIT,IBADMODE,ITIMEOUT,
52   0    1:0     1                  ILOSTUNIT,ILOSTFILE,IBADTITLE,INOROOM,INOUNIT,
53   0    1:0     1                  INOFILE,IDUPFILE,INOTCLOSED,INOTOPEN,IBADFORMAT,
54   0    1:0     1                  ISTRGOVFL);
55   0    1:0     1
56   0    1:0     1                                          (*COMMAND STATES...SEE GETCMD*)
57   0    1:0     1
58   0    1:0     1      CMDSTATE = (HALTINIT,DEBUGCALL,
59   0    1:0     1                  UPROGNOU,UPROGUOK,SYSPROG,
60   0    1:0     1                  COMPONLY,COMPANDGO,COMPDEBUG,
61   0    1:0     1                  LINKANDGO,LINKDEBUG);
62   0    1:0     1
63   0    1:0     1                                          (*CODE FILES USED IN GETCMD*)
64   0    1:0     1
65   0    1:0     1      SYSFILE = (ASSMBLER,COMPILER,EDITOR,FILER,LINKER);
66   0    1:0     1
67   0    1:0     1                                          (*ARCHIVAL INFO...THE DATE*)
68   0    1:0     1
69   0    1:0     1      DATEREC = PACKED RECORD
70   0    1:0     1                  MONTH: 0..12;          (*0 IMPLIES DATE NOT MEANINGFUL*)
71   0    1:0     1                  DAY: 0..31;            (*DAY OF MONTH*)
72   0    1:0     1                  YEAR: 0..100           (*100 IS TEMP DISK FLAG*)
73   0    1:0     1                END (*DATEREC*) ;
74   0    1:0     1
75   0    1:0     1                                          (*VOLUME TABLES*)
76   0    1:0     1      UNITNUM = 0..MAXUNIT;
77   0    1:0     1      VID = STRING[VIDLENG];
78   0    1:0     1
79   0    1:0     1                                          (*DISK DIRECTORIES*)
80   0    1:0     1      DIRRANGE = 0..MAXDIR;
81   0    1:0     1      TID = STRING[TIDLENG];
82   0    1:0     1      FULL_ID = STRING[NAME_LEN];
83   0    1:0     1
84   0    1:0     1      FILE_TABLE = ARRAY [SYSFILE] OF FULL_ID;
85   0    1:0     1
86   0    1:0     1      FILEKIND = (UNTYPEDFILE,[illegible]KFILE,CODEFILE,TEXTFILE,
```

```
 87   0    1:0     1                INFOFILE,DATAFILE,GRAFFILE,FOTOFILE,SECUREDIR);
 88   0    1:0     1
 89   0    1:0     1      DIRENTRY = PACKED RECORD
 90   0    1:0     1                   DFIRSTBLK: INTEGER;    (*FIRST PHYSICAL DISK ADDR*)
 91   0    1:0     1                   DLASTBLK: INTEGER;     (*POINTS AT BLOCK FOLLOWING*)
 92   0    1:0     1                   CASE DFKIND: FILEKIND OF
 93   0    1:0     1                     SECUREDIR,
 94   0    1:0     1                     UNTYPEDFILE: (*ONLY IN DIR[0]...VOLUME INFO*)
 95   0    1:0     1                        (FILLER1 : 0..2048; [FOR DOWNWARD COMPATIBILITY,13 BITS]
 96   0    1:0     1                         DVID: VID;                (*NAME OF DISK VOLUME*)
 97   0    1:0     1                         DEOVBLK: INTEGER;         (*LASTBLK OF VOLUME*)
 98   0    1:0     1                         DNUMFILES: DIRRANGE;      (*NUM FILES IN DIR*)
 99   0    1:0     1                         DLOADTIME: INTEGER;       (*TIME OF LAST ACCESS*)
100   0    1:0     1                         DLASTBOOT: DATEREC);      (*MOST RECENT DATE SETTING*)
101   0    1:0     1                     XDSKFILE,CODEFILE,TEXTFILE,INFOFILE,
102   0    1:0     1                     DATAFILE,GRAFFILE,FOTOFILE:
103   0    1:0     1                        (FILLER2 : 0..1024; [FOR DOWNWARD COMPATIBILITY]
104   0    1:0     1                         STATUS : BOOLEAN;          [FOR FILER WILDCARDS]
105   0    1:0     1                         DTID: TID;                (*TITLE OF FILE*)
106   0    1:0     1                         DLASTBYTE: 1..FBLKSIZE;   (*NUM BYTES IN LAST BLOCK*)
107   0    1:0     1                         DACCESS: DATEREC)         (*LAST MODIFICATION DATE*)
108   0    1:0     1                END (*DIRENTRY*) ;
109   0    1:0     1
110   0    1:0     1      DIRP = ^DIRECTORY;
111   0    1:0     1
112   0    1:0     1      DIRECTORY = ARRAY [DIRRANGE] OF DIRENTRY;
113   0    1:0     1
114   0    1:0     1                                              (*FILE INFORMATION*)
115   0    1:0     1
116   0    1:0     1      CLOSETYPE = (CNORMAL,CLOCK,CPURGE,CCRUNCH);
117   0    1:0     1      WINDOWP = ^WINDOW;
118   0    1:0     1      WINDOW = PACKED ARRAY [0..0] OF CHAR;
119   0    1:0     1      FIBP = ^FIB;
120   0    1:0     1
121   0    1:0     1      FIB = RECORD
122   0    1:0     1              FWINDOW: WINDOWP;  (*USER WINDOW...F^, USED BY GET-PUT*)
123   0    1:0     1              FEOF,FEOLN: BOOLEAN;
124   0    1:0     1              FSTATE: (FJANDW,FNEEDCHAR,FGOTCHAR);
125   0    1:0     1              FRECSIZE: INTEGER; (*IN BYTES...0=>BLOCKFILE, 1=>CHARFILE*)
126   0    1:0     1              CASE FISOPEN: BOOLEAN OF
127   0    1:0     1                TRUE: (FISBLKD: BOOLEAN; (*FILE IS ON BLOCK DEVICE*)
```

```
128   0    1:D    1                    FUNIT: UNITNUM;    (*PHYSICAL UNIT #*)
129   0    1:D    1                    FVID: VID;         (*VOLUME NAME*)
130   0    1:D    1                    FREPTCNT,          (* # TIMES F^ VALID W/O GET*)
131   0    1:D    1                    FNXTBLK,           (*NEXT REL BLOCK TO IO*)
132   0    1:D    1                    FMAXBLK: INTEGER;  (*MAX REL BLOCK ACCESSED*)
133   0    1:D    1                    FMODIFIED:BOOLEAN;(*PLEASE SET NEW DATE IN CLOSE*)
134   0    1:D    1                    FHEADER: DIRENTRY;(*COPY OF DISK DIR ENTRY*)
135   0    1:D    1                    CASE FSOFTBUF: BOOLEAN OF (*DISK GET-PUT STUFF*)
136   0    1:D    1                      TRUE: (FNXTBYTE,FMAXBYTE: INTEGER;
137   0    1:D    1                             FBUFCHNGD: BOOLEAN;
138   0    1:D    1                             FBUFFER: PACKED ARRAY [0..FBLKSIZE] OF CHAR))
139   0    1:D    1            END (*FIB*) ;
140   0    1:D    1
141   0    1:D    1                                        (*USER WORKFILE STUFF*)
142   0    1:D    1
143   0    1:D    1        INFOREC = RECORD
144   0    1:D    1                    SYMFIBP,CODEFIBP: FIBP;          (*WORKFILES FOR SCRATCH*)
145   0    1:D    1                    ERRSYM,ERRBLK,ERRNUM: INTEGER;   (*ERROR STUFF IN EDIT*)
146   0    1:D    1                    SLOWTERM,STUPID: BOOLEAN;        (*STUDENT PROGRAMMER ID!!*)
147   0    1:D    1                    ALTMODE: CHAR;                   (*WASHOUT CHAR FOR COMPILER*)
148   0    1:D    1                    GOTSYM,GOTCODE: BOOLEAN;         (*TITLES ARE MEANINGFUL*)
149   0    1:D    1                    WORKVID,SYMVID,CODEVID: VID;     (*PERM&CUR WORKFILE VOLUMES*)
150   0    1:D    1                    WORKTID,SYMTID,CODETID: TID      (*PERM&CUR WORKFILES TITLE*)
151   0    1:D    1                  END (*INFOREC*) ;
152   0    1:D    1
153   0    1:D    1                                        (*CODE SEGMENT LAYOUTS*)
154   0    1:D    1
155   0    1:D    1        SEGRANGE = 0..MAXSEG;
156   0    1:D    1        SEGDESC = RECORD
157   0    1:D    1                    DISKADDR: INTEGER;       (*REL BLK IN CODE...ABS IN SYSCOM^*)
158   0    1:D    1                    CODELENG: INTEGER        (*# BYTES TO READ IN*)
159   0    1:D    1                  END (*SEGDESC*) ;
160   0    1:D    1
161   0    1:D    1                                        (*DEBUGGER STUFF*)
162   0    1:D    1
163   0    1:D    1        BYTERANGE = 0..255;
164   0    1:D    1        TRICKARRAY = RECORD          [MEMORY DIDDLING FOR EXECERROR]
165   0    1:D    1                      CASE BOOLEAN OF
166   0    1:D    1                        TRUE : (WORD : ARRAY [0..0] OF INTEGER);
167   0    1:D    1                        FALSE : (BYTE : PACKED ARRAY [0..0] OF BYTERANGE)
168   0    1:D    1                      END;
```

```
169   0     1:0      1      MSCWP = ^ MSCW;                (*MARK STACK RECORD POINTER*)
170   0     1:0      1      MSCW = RECORD
171   0     1:0      1               STATLINK: MSCWP;   (*POINTER TO PARENT MSCW*)
172   0     1:0      1               DYNLINK: MSCWP;    (*POINTER TO CALLER'S MSCW*)
173   0     1:0      1               MSSEG,MSJTAB: ^TRICKARRAY;
174   0     1:0      1               MSIPC: INTEGER;
175   0     1:0      1               LOCALDATA: TRICKARRAY
176   0     1:0      1             END (*MSCW*) ;
177   0     1:0      1
178   0     1:0      1                                          (*SYSTEM COMMUNICATION AREA*)
179   0     1:0      1                                          (*SEE INTERPRETERS...NOTE  *)
180   0     1:0      1                                          (*THAT WE ASSUME BACKWARD  *)
181   0     1:0      1                                          (*FIELD ALLOCATION IS DONE *)
182   0     1:0      1      SYSCOMREC = RECORD
183   0     1:0      1                   IORSLT: IORSLTWD;     (*RESULT OF LAST IO CALL*)
184   0     1:0      1                   XEQERR: INTEGER;      (*REASON FOR EXECERROR CALL*)
185   0     1:0      1                   SYSUNIT: UNITNUM;     (*PHYSICAL UNIT OF BOOTLOAD*)
186   0     1:0      1                   BUGSTATE: INTEGER;    (*DEBUGGER INFO*)
187   0     1:0      1                   GDIRP: DIRP;          (*GLOBAL DIR POINTER,SEE VOLSEARCH*)
188   0     1:0      1                   LASTMP,STKBASE,BOMBP: MSCWP;
189   0     1:0      1                   MEMTOP,SEG,JTAB: INTEGER;
190   0     1:0      1                   BOMBIPC: INTEGER;     (*WHERE XEQERR BLOWUP WAS*)
191   0     1:0      1                   HLTLINE: INTEGER;     (*MORE DEBUGGER STUFF*)
192   0     1:0      1                   BRKPTS: ARRAY [0..3] OF INTEGER;
193   0     1:0      1                   RETRIES: INTEGER;     (*DRIVERS PUT RETRY COUNTS*)
194   0     1:0      1                   EXPANSION: ARRAY [0..8] OF INTEGER;
195   0     1:0      1                   HIGHTIME,LOWTIME: INTEGER;
196   0     1:0      1                   MISCINFO: PACKED RECORD
197   0     1:0      1                               NOBREAK,STUPID,SLOWTERM,
198   0     1:0      1                               HASXYCRT,HASLCCRT,HAS8510A,HASCLOCK: BOOLEAN;
199   0     1:0      1                               USERKIND:(NORMAL, AQUIZ, BOOKER, PQUIZ);
200   0     1:0      1                               IS_FLIPT : BOOLEAN
201   0     1:0      1                             END;
202   0     1:0      1                   CRTTYPE: INTEGER;
203   0     1:0      1                   CRTCTRL: PACKED RECORD
204   0     1:0      1                              RLF,NDFS,ERASEEOL,ERASEEOS,HOME,ESCAPE: CHAR;
205   0     1:0      1                              BACKSPACE: CHAR;
206   0     1:0      1                              FILLCOUNT: 0..255;
207   0     1:0      1                              CLEARSCREEN, CLEARLINE: CHAR;
208   0     1:0      1                              PREFIXED: PACKED ARRAY [0..8] OF BOOLEAN
209   0     1:0      1                            END;
```

```
210   0    1:D     1                    CRTINFO: PACKED RECORD
211   0    1:D     1                               WIDTH,HEIGHT: INTEGER;
212   0    1:D     1                               RIGHT,LEFT,DOWN,UP: CHAR;
213   0    1:D     1                               BADCH,CHARDEL,STOP,BREAK,FLUSH,EOF: CHAR;
214   0    1:D     1                               ALTMODE,LINEDEL: CHAR;
215   0    1:D     1                               BACKSPACE,ETX,PREFIX: CHAR;
216   0    1:D     1                               PREFIXED: PACKED ARRAY [0..13] OF BOOLEAN
217   0    1:D     1                             END;
218   0    1:D     1                    SEGTABLE: ARRAY [SEGRANGE] OF
219   0    1:D     1                                RECORD
220   0    1:D     1                                  CODEUNIT: UNITNUM;
221   0    1:D     1                                  CODEDESC: SEGDESC
222   0    1:D     1                                END
223   0    1:D     1                  END (*SYSCOM*);
224   0    1:D     1
225   0    1:D     1      MISCINFOREC = RECORD
226   0    1:D     1                      MSYSCOM: SYSCOMREC
227   0    1:D     1                    END;
228   0    1:D     1
229   0    1:D     1  VAR
230   0    1:D     1      SYSCOM: ^SYSCOMREC;                    (*MAGIC PARAM...SET UP IN BOOT*)
231   0    1:D     2      GFILES: ARRAY [0..5] OF FIBP;          (*GLOBAL FILES, 0=INPUT, 1=OUTPUT*)
232   0    1:D     8      USERINFO: INFOREC;                     (*WORK STUFF FOR COMPILER ETC*)
233   0    1:D    54      EMPTYHEAP: ^INTEGER;                   (*HEAP MARK FOR MEM MANAGING*)
234   0    1:D    55      INPUTFIB,OUTPUTFIB,                    (*CONSOLE FILES...GFILES ARE COPIES*)
235   0    1:D    55      SYSTERM,SWAPFIB: FIBP;                 (*CONTROL AND SWAPSPACE FILES*)
236   0    1:D    59      SYVID,DKVID: VID;                      (*SYSUNIT VOLID & DEFAULT VOLID*)
237   0    1:D    67      THEDATE: DATEREC;                      (*TODAY...SET IN FILER OR SIGN ON*)
238   0    1:D    68      DEBUGINFO: ^INTEGER;                   (*DEBUGGERS GLOBAL INFO WHILE RUNIN*)
239   0    1:D    69      STATE: CMDSTATE;                       (*FOR GETCOMMAND*)
240   0    1:D    70      PL: STRING;                            (*PROMPTLINE STRING...SEE PROMPT*)
241   0    1:D   111      IPOT: ARRAY [0..4] OF INTEGER;         (*INTEGER POWERS OF TEN*)
242   0    1:D   116      FILLER: STRING[FILL_LEN];              (*NULLS FOR CARRIAGE DELAY*)
243   0    1:D   122      DIGITS: SET OF '0'..'9';
244   0    1:D   126      UNITABLE: ARRAY [UNITNUM] OF (*0 NOT USED*)
245   0    1:D   126                  RECORD
246   0    1:D   126                    UVID: VID;    (*VOLUME ID FOR UNIT*)
247   0    1:D   126                    CASE UISBLKD: BOOLEAN OF
248   0    1:D   126                      TRUE: (UEOVBLK: INTEGER)
249   0    1:D   126                  END (*UNITABLE*) ;
250   0    1:D   204      FILENAME : FILE_TABLE;
```

```
251   0     1:D   264
252   0     1:D   264 (*-------------------------------------------------------------------------------*)
253   0     1:D   264 (* SYSTEM PROCEDURE FORWARD DECLARATIONS *)
254   0     1:D   264 (* THESE ARE ADDRESSED BY OBJECT CODE... *)
255   0     1:D   264 (*  DO NOT MOVE WITHOUT CAREFUL THOUGHT  *)
256   0     1:D   264
257   0     2:D     1 PROCEDURE EXECERROR;
258   0     2:D     1   FORWARD;
259   0     3:D     1 PROCEDURE FINIT(VAR F: FIB; WINDOW: WINDOWP; RECWORDS: INTEGER);
260   0     3:D     4   FORWARD;
261   0     4:D     1 PROCEDURE FRESET(VAR F: FIB);
262   0     4:D     2   FORWARD;
263   0     5:D     1 PROCEDURE FOPEN(VAR F: FIB; VAR FTITLE: STRING;
264   0     5:D     3                   FOPENOLD: BOOLEAN; JUNK: FIBP);
265   0     5:D     5   FORWARD;
266   0     6:D     1 PROCEDURE FCLOSE(VAR F: FIB; FTYPE: CLOSETYPE);
267   0     6:D     3   FORWARD;
268   0     7:D     1 PROCEDURE FGET(VAR F: FIB);
269   0     7:D     2   FORWARD;
270   0     8:D     1 PROCEDURE FPUT(VAR F: FIB);
271   0     8:D     2   FORWARD;
272   0     9:D     1 PROCEDURE XSEEK;
273   0     9:D     1   FORWARD;
274   0    10:D     3 FUNCTION FEOF(VAR F: FIB): BOOLEAN;
275   0    10:D     4   FORWARD;
276   0    11:D     3 FUNCTION FEOLN(VAR F: FIB): BOOLEAN;
277   0    11:D     4   FORWARD;
278   0    12:D     1 PROCEDURE FREADINT(VAR F: FIB; VAR I: INTEGER);
279   0    12:D     3   FORWARD;
280   0    13:D     1 PROCEDURE FWRITEINT(VAR F: FIB; I,RLENG: INTEGER);
281   0    13:D     4   FORWARD;
282   0    14:D     1 PROCEDURE XREADREAL;
283   0    14:D     1   FORWARD;
284   0    15:D     1 PROCEDURE XWRITEREAL;
285   0    15:D     1   FORWARD;
286   0    16:D     1 PROCEDURE FREADCHAR(VAR F: FIB; VAR CH: CHAR);
287   0    16:D     3   FORWARD;
288   0    17:D     1 PROCEDURE FWRITECHAR(VAR F: FIB; CH: CHAR; RLENG: INTEGER);
289   0    17:D     4   FORWARD;
290   0    18:D     1 PROCEDURE FREADSTRING(VAR F: FIB; VAR S: STRING; SLENG: INTEGER);
291   0    18:D     4   FORWARD;
```

```
292   0   19:0    1 PROCEDURE FWRITESTRING(VAR F: FIB; VAR S: STRING; RLENG: INTEGER);
293   0   19:0    4   FORWARD;
294   0   20:0    1 PROCEDURE FWRITEBYTES(VAR F: FIB; VAR A: WINDOW; RLENG,ALENG: INTEGER);
295   0   20:0    5   FORWARD;
296   0   21:0    1 PROCEDURE FREADLN(VAR F: FIB);
297   0   21:0    2   FORWARD;
298   0   22:0    1 PROCEDURE FWRITELN(VAR F: FIB);
299   0   22:0    2   FORWARD;
300   0   23:0    1 PROCEDURE SCONCAT(VAR DEST,SRC: STRING; DESTLENG: INTEGER);
301   0   23:0    4   FORWARD;
302   0   24:0    1 PROCEDURE SINSERT(VAR SRC,DEST: STRING; DESTLENG,INSINX: INTEGER);
303   0   24:0    5   FORWARD;
304   0   25:0    1 PROCEDURE SCOPY(VAR SRC,DEST: STRING; SRCINX,COPYLENG: INTEGER);
305   0   25:0    5   FORWARD;
306   0   26:0    1 PROCEDURE SDELETE(VAR DEST: STRING; DELINX,DELLENG: INTEGER);
307   0   26:0    4   FORWARD;
308   0   27:0    3 FUNCTION SPOS(VAR TARGET,SRC: STRING): INTEGER;
309   0   27:0    5   FORWARD;
310   0   28:0    3 FUNCTION FBLOCKIO(VAR F: FIB; VAR A: WINDOW; I: INTEGER;
311   0   28:0    6                       NBLOCKS,RBLOCK: INTEGER; DOREAD: BOOLEAN): INTEGER;
312   0   28:0    9   FORWARD;
313   0   29:0    1 PROCEDURE FGOTOXY(X,Y: INTEGER);
314   0   29:0    3   FORWARD;
315   0   29:0    3
316   0   29:0    3 (* NON FIXED FORWARD DECLARATIONS *)
317   0   29:0    3
318   0   30:0    3 FUNCTION VOLSEARCH(VAR FVID: VID; LOOKHARD: BOOLEAN;
319   0   30:0    5                       VAR FDIR: DIRP): UNITNUM;
320   0   30:0    6   FORWARD;
321   0   31:0    1 PROCEDURE WRITEDIR(FUNIT: UNITNUM; FDIR: DIRP);
322   0   31:0    3   FORWARD;
323   0   32:0    3 FUNCTION DIRSEARCH(VAR FTID: TID; FINDPERM: BOOLEAN; FDIR: DIRP): DIRRANGE;
324   0   32:0    6   FORWARD;
325   0   33:0    3 FUNCTION SCANTITLE(FTITLE: STRING; VAR FVID: VID; VAR FTID: TID;
326   0   33:0    6                       VAR FSEGS: INTEGER; VAR FKIND: FILEKIND): BOOLEAN;
327   0   33:0   49   FORWARD;
328   0   34:0    1 PROCEDURE DELENTRY(FINX: DIRRANGE; FDIR: DIRP);
329   0   34:0    3   FORWARD;
330   0   35:0    1 PROCEDURE INSENTRY(VAR FENTRY: DIRENTRY; FINX: DIRRANGE; FDIR: DIRP);
331   0   35:0    4   FORWARD;
332   0   36:0    1 PROCEDURE HOMECURSOR;
```

```
333   0    36:0     1    FORWARD;
334   0    37:0     1  PROCEDURE CLEARSCREEN;
335   0    37:0     1    FORWARD;
336   0    38:0     1  PROCEDURE CLEARLINE;
337   0    38:0     1    FORWARD;
338   0    39:0     1  PROCEDURE PROMPT;
339   0    39:0     1    FORWARD;
340   0    40:0     3  FUNCTION SPACEWAIT(FLUSH: BOOLEAN): BOOLEAN;
341   0    40:0     4    FORWARD;
342   0    41:0     3  FUNCTION GETCHAR(FLUSH: BOOLEAN): CHAR;
343   0    41:0     4    FORWARD;
344   0    42:0     3  FUNCTION FETCHDIR(FUNIT:UNITNUM) : BOOLEAN;
345   0    42:0     4    FORWARD;
346   0    43:0     1  PROCEDURE COMMAND;
347   0    43:0     1    FORWARD;
348   0    43:0     1
349   0    43:0     1
350   0    43:0     1  [$I GLOBALS ]
350   0    43:0     1  [$I SYSSEGS.A  ]
351   0    43:0     1
352   0    43:0     1      (*****************************************************************)
353   0    43:0     1      (*                                                               *)
354   0    43:0     1      (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
355   0    43:0     1      (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
356   0    43:0     1      (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
357   0    43:0     1      (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
358   0    43:0     1      (*                                                               *)
359   0    43:0     1      (*****************************************************************)
360   0    43:0     1
361   1     1:0     1  SEGMENT PROCEDURE USERPROGRAM(INPUT,OUTPUT: FIBP);
362   1     1:0     0  BEGIN FWRITELN(SYSTERM^);
363   1     1:1     6    PL := 'NO USER PROGRAM';
364   1     1:1    29    FWRITESTRING(SYSTERM^,PL,0)
365   1     1:0    36  END (*USERPROGRAM*) ;
366   1     1:0    52
367   2     1:0     1  SEGMENT PROCEDURE DEBUGGER;
368   2     1:0     0  BEGIN FWRITELN(SYSTERM^);
369   2     1:1     6    PL := 'NO DEBUGGER IN SYSTEM';
370   2     1:1    35    FWRITESTRING(SYSTERM^,PL,0)
371   2     1:0    42  END (*DEBUGGER*) ;
372   2     1:0    58
```

```
373   3    1:0      1 SEGMENT PROCEDURE PRINTERROR(XEQERR,IORSLT: INTEGER);
374   3    1:0      3   VAR S: STRING[40];
375   3    1:0      0 BEGIN S := 'UNKNOWN RUN-TIME ERROR';
376   3    1:1     29   CASE XEQERR OF
377   3    1:1     32     1:  S := 'VALUE RANGE ERROR';
378   3    1:1     58     2:  S := 'NO PROC IN SEG-TABLE';
379   3    1:1     87     3:  S := 'EXIT FROM UNCALLED PROC';
380   3    1:1    119     4:  S := 'STACK OVERFLOW';
381   3    1:1    142     5:  S := 'INTEGER OVERFLOW';
382   3    1:1    167     6:  S := 'DIVIDE BY ZERO';
383   3    1:1    190     7:  S := 'NIL POINTER REFERENCE';
384   3    1:1    220     8:  S := 'PROGRAM INTERRUPTED BY USER';
385   3    1:1    256     9:  S := 'SYSTEM IO ERROR';
386   3    1:1    280    10:  BEGIN S := 'UNKNOWN CAUSE';
387   3    1:3    300           CASE IORSLT OF
388   3    1:3    303             1:  S := 'PARITY (CRC)';
389   3    1:3    324             2:  S := 'ILLEGAL UNIT #';
390   3    1:3    347             3:  S := 'ILLEGAL IO REQUEST';
391   3    1:3    374             4:  S := 'DATA-COM TIMEOUT';
392   3    1:3    399             5:  S := 'VOL WENT OFF-LINE';
393   3    1:3    425             6:  S := 'FILE LOST IN DIR';
394   3    1:3    450             7:  S := 'BAD FILE NAME';
395   3    1:3    472             8:  S := 'NO ROOM ON VOL';
396   3    1:3    495             9:  S := 'VOL NOT FOUND';
397   3    1:3    517            10:  S := 'FILE NOT FOUND';
398   3    1:3    540            11:  S := 'DUP DIR ENTRY';
399   3    1:3    562            12:  S := 'FILE ALREADY OPEN';
400   3    1:3    588            13:  S := 'FILE NOT OPEN';
401   3    1:3    610            14:  S := 'BAD INPUT FORMAT';
402   3    1:3    635            15:  S := 'RING BUFFER OVERFLOW';
403   3    1:3    664            16:  S := 'DISK WRITE PROTECTED';
404   3    1:3    693            17:  S := 'ILLEGAL BLOCK #';
405   3    1:3    717            18:  S := 'BAD BYTE COUNT';
406   3    1:3    740            19:  S := 'BAD INIT RECORD'
407   3    1:3    742           END (*IO ERRORS*) ;
408   3    1:3    810         INSERT('IO ERROR: ',S,1)
409   3    1:2    830       END;
410   3    1:1    832    11:  S := 'UNIMPLEMENTED INSTRUCTION';
411   3    1:1    866    12:  S := 'FLOATING POINT ERROR';
412   3    1:1    895    13:  S := 'STRING OVERFLOW';
413   3    1:1    919    14:  S := 'PROGRAMMED HALT'
```

```
414   3     1:1    943     15:  S := 'PROGRAMMED BREAK-POINT'
415   3     1:1    945   END (*XEQ ERRORS*) ;
416   3     1:1   1012   WRITELN(OUTPUT,S);
417   3     1:0   1027 END (*PRINTERROR*) ;
418   3     1:0   1048
419   4     1:0      1 SEGMENT PROCEDURE INITIALIZE;
420   4     1:0      1   VAR JUSTBOOTED: BOOLEAN; LTITLE: STRING[40];
421   4     1:0     23       MONTHS: ARRAY [0..15] OF STRING[3];
422   4     1:0     55       STARTUP : BOOLEAN;
423   4     1:0     56       STKFILL: ARRAY [0..1199] OF INTEGER;
424   4     1:0   1256
425   4     2:0      1   PROCEDURE INITSYSCOM;
426   4     2:0      1     VAR TITLE: STRING;
427   4     2:0     42         F: FILE OF MISCINFOREC;
428   4     2:0    438
429   4     3:0      1     PROCEDURE INIT_FILLER(VAR  FILLER : STRING);
430   4     3:0      0       BEGIN
431   4     3:1      0         WITH SYSCOM^.CRTCTRL DO
432   4     3:2      7           BEGIN
433   4     3:3      7             IF FILLCOUNT > FILL_LEN THEN
434   4     3:4     17               FILLCOUNT := FILL_LEN;
435   4     3:3     24             FILLER[0] := CHR(FILLCOUNT);
436   4     3:3     33             FILLCHAR(FILLER[1],FILLCOUNT,CHR(0));
437   4     3:2     44             END;
438   4     3:0     44       END  [OF INIT_FILLER];
439   4     3:0     56
440   4     2:0      0     BEGIN                 [OF INITSYSCOM]
441   4     2:1      0       INIT_FILLER(FILLER);
442   4     2:1     14       DEBUGINFO := NIL;
443   4     2:1     18       IPOT[0] := 1; IPOT[1] := 10; IPOT[2] := 100;
444   4     2:1     42       IPOT[3] := 1000; IPOT[4] := 10000; DIGITS := ['0'..'9'];
445   4     2:1     81       WITH SYSCOM^ DO
446   4     2:2     87         BEGIN
447   4     2:3     87         XEQERR := 0;     IORSLT := INOERROR;
448   4     2:3     99         BUGSTATE :=0
449   4     2:2    104         END;
450   4     2:1    106       TITLE := '*SYSTEM.MISCINFO' ;
451   4     2:1    129       RESET( F, TITLE );
452   4     2:1    138       IF IORESULT = ORD(INOERROR) THEN
453   4     2:2    144         BEGIN
454   4     2:3    144         IF NOT EOF( F ) THEN
```

```
455   4    2:4   154          WITH SYSCOM^, F^ DO
456   4    2:5   165            BEGIN
457   4    2:6   165            MISCINFO := MSYSCOM.MISCINFO;
458   4    2:6   177            CRTTYPE := MSYSCOM.CRTTYPE;
459   4    2:6   188            CRTCTRL := MSYSCOM.CRTCTRL;
460   4    2:6   200            CRTINFO := MSYSCOM.CRTINFO;
461   4    2:6   212            INIT_FILLER(FILLER);
462   4    2:5   217            END;
463   4    2:3   217        CLOSE( F, NORMAL )
464   4    2:2   223        END;
465   4    2:1   223      UNITCLEAR(1) (*GIVE BIOS NEW SOFT CHARACTERS FOR CONSOLE*)
466   4    2:0   224    END (*INITSYSCOM*) ;
467   4    2:0   244
468   4    4:0     1  PROCEDURE INITUNITABLE;
469   4    4:0     1  VAR LUNIT: UNITNUM;
470   4    4:0     2      LDIR: DIRP;
471   4    4:0     3      LFIB : FIB;
472   4    4:0   293      F : SYSFILE;
473   4    4:0   294      TEMP_NAMES : FILE_TABLE;
474   4    4:0   354      NOT_FOUND : SET OF SYSFILE;
475   4    4:0   355
476   4    5:0     1    PROCEDURE INIT_ENTRY(LUNIT : UNITNUM; UNIT_NAME : VID);
477   4    5:0     0    BEGIN
478   4    5:1     0      UNITCLEAR(LUNIT);
479   4    5:1     8      IF IORESULT = ORD(INOERROR) THEN
480   4    5:2    14        UNITABLE[LUNIT].UVID := UNIT_NAME;
481   4    5:0    24      END  [OF INIT_ENTRY];
482   4    5:0    36
483   4    4:0     0  BEGIN          [OF INITUNITABLE]
484   4    4:1     0    FILENAME[ASSMBLER] := ':SYSTEM.ASSMBLER';
485   4    4:1    28    FILENAME[COMPILER] := ':SYSTEM.COMPILER';
486   4    4:1    56    FILENAME[EDITOR] := ':SYSTEM.EDITOR';
487   4    4:1    82    FILENAME[FILER] := ':SYSTEM.FILER';
488   4    4:1   107    FILENAME[LINKER] := ':SYSTEM.LINKER';
489   4    4:1   133    TEMP_NAMES := FILENAME;
490   4    4:1   142    NOT_FOUND := [ASSMBLER .. LINKER];
491   4    4:1   149    FINIT(LFIB,NIL,-1);
492   4    4:1   157    FOR LUNIT := 0 TO MAXUNIT DO
493   4    4:2   171      WITH UNITABLE[LUNIT] DO
494   4    4:3   180        BEGIN
495   4    4:4   180          UVID := '';
```

```
496   4     4:4    183          UISBLKD := LUNIT IN [4,5,9..12];
497   4     4:4    200          IF UISBLKD THEN
498   4     4:5    206            BEGIN
499   4     4:6    206              UEOVBLK := MMAXINT;
500   4     4:6    215              UNITCLEAR(LUNIT);
501   4     4:6    218              IF IORESULT = ORD(INOERROR) THEN
502   4     4:7    224                IF FETCHDIR(LUNIT) THEN
503   4     4:8    232                  BEGIN
504   4     4:9    232                    UVID := SYSCOM^.GDIRP^[0].DVID;
505   4     4:9    246                    IF LUNIT = SYSCOM^.SYSUNIT THEN
506   4     4:0    254                      BEGIN
507   4     4:1    254                        SYVID := UVID;
508   4     4:1    262                        LTITLE := '*SYSTEM.STARTUP';
509   4     4:1    284                        FOPEN(LFIB,LTITLE,TRUE,NIL);
510   4     4:1    293                        STARTUP := LFIB.FISOPEN;
511   4     4:1    296                        FCLOSE(LFIB,CNORMAL);
512   4     4:0    302                        END;
513   4     4:9    302                    FOR F := ASSMBLER TO LINKER DO
514   4     4:0    319                      IF (LUNIT = SYSCOM^.SYSUNIT) OR (F IN NOT_FOUND) THEN
515   4     4:1    336                        BEGIN
516   4     4:2    336                          LTITLE := CONCAT(UVID,TEMP_NAMES[F]);
517   4     4:2    372                          FOPEN(LFIB,LTITLE,TRUE,NIL);
518   4     4:2    381                          IF LFIB.FISOPEN THEN
519   4     4:3    384                            BEGIN
520   4     4:4    384                              FILENAME[F] := LTITLE;
521   4     4:4    397                              NOT_FOUND := NOT_FOUND - [F];
522   4     4:3    411                              END;
523   4     4:2    411                          FCLOSE(LFIB,CNORMAL);
524   4     4:1    417                          END [OF IF (LUNIT ...];
525   4     4:8    427                    END [OF IF FETCHDIR .. ] ;
526   4     4:5    427                END  [OF IF UISBLKD .. ];
527   4     4:3    427              END [OF WITH];
528   4     4:1    434        IF JUSTBOOTED THEN
529   4     4:2    437          DKVID := SYVID;
530   4     4:1    445        LUNIT := VOLSEARCH(SYVID,FALSE,LDIR);
531   4     4:1    458        IF LDIR = NIL THEN
532   4     4:2    463          HALT;
533   4     4:1    465        THEDATE := LDIR^[0].DLASTBOOT;
534   4     4:1    476        INIT_ENTRY(1,'CONSOLE');
535   4     4:1    489        INIT_ENTRY(2,'SYSTERM');
536   4     4:1    502        INIT_ENTRY(3,'GRAPHIC');
```

```
537   4    4:1   515     INIT_ENTRY(6,'PRINTER');
538   4    4:1   528     INIT_ENTRY(7,'REMIN');
539   4    4:1   539     INIT_ENTRY(8,'REMOUT');
540   4    4:0   551     END [OF INITUNITABLE];
541   4    4:0   576
542   4    6:D     1   PROCEDURE INITCHARSET;
543   4    6:D     1   TYPE CHARSET= ARRAY [32..127] OF
544   4    6:D     1                    PACKED ARRAY [0..9] OF 0..255;
545   4    6:D     1   VAR I: INTEGER;
546   4    6:D     2       DOTRITON : BOOLEAN;
547   4    6:D     3       TRIX: RECORD CASE BOOLEAN OF
548   4    6:D     3               TRUE:  (CHARADDR: INTEGER);
549   4    6:D     3               FALSE: (CHARBUFP: ^ CHAR)
550   4    6:D     3             END;
551   4    6:D     4       DISPLAY: ARRAY [0..79,0..19] OF INTEGER; (*FOR TRITON*)
552   4    6:D  1604       CHARBUF: RECORD
553   4    6:D  1604                  SET1: CHARSET;
554   4    6:D  1604                  FILLER1: PACKED ARRAY [0..63] OF CHAR;
555   4    6:D  1604                  SET2: CHARSET;
556   4    6:D  1604                  FILLER2: PACKED ARRAY [0..63] OF CHAR;
557   4    6:D  1604                  TRITON: ARRAY [0..63,0..3] OF INTEGER
558   4    6:D  1604                END (*CHARBUF*) ;
559   4    6:D  2884       LFIB: FIB;
560   4    6:0     0   BEGIN FINIT(LFIB,NIL,-1);
561   4    6:1     9     LTITLE := '*SYSTEM.CHARSET';
562   4    6:1    31     FOPEN(LFIB,LTITLE,TRUE,NIL);
563   4    6:1    41     IF LFIB.FISOPEN THEN
564   4    6:2    46       BEGIN
565   4    6:3    46         UNITCLEAR(3);
566   4    6:3    49         IF IORESULT = ORD(INOERROR) THEN
567   4    6:4    55           BEGIN
568   4    6:5    55             UNITWRITE(3,TRIX,128);
569   4    6:5    66             WITH LFIB.FHEADER DO
570   4    6:6    66               BEGIN
571   4    6:7    66                 DOTRITON := DLASTBLK-DFIRSTBLK > 4;
572   4    6:7    77                 UNITREAD(LFIB.FUNIT,CHARBUF,SIZEOF(CHARBUF),DFIRSTBLK)
573   4    6:6    93               END;
574   4    6:5    93             TRIX.CHARADDR := 512-8192;  (*UNIBUS TRICKYNESS!*)
575   4    6:5   102             FOR I := 32 TO 127 DO
576   4    6:6   116               BEGIN
577   4    6:7   116                 MOVERIGHT(CHAR 'F.SET1[I],TRIX.CHARBUFP^,10);
```

```
578   4    6:7   150                TRIX.CHARADDR := TRIX.CHARADDR+16
579   4    6:6   131              END;
580   4    6:5   142            TRIX.CHARADDR := 512-6144;
581   4    6:5   151            FOR I := 32 TO 127 DO
582   4    6:6   165              BEGIN
583   4    6:7   165                MOVERIGHT(CHARBUF.SET2[I],TRIX.CHARBUFP^,10);
584   4    6:7   179                TRIX.CHARADDR := TRIX.CHARADDR+16
585   4    6:6   180              END;
586   4    6:5   191            IF JUSTBOOTED AND DOTRITON AND NOT STARTUP THEN
587   4    6:6   200              BEGIN (*INITIALIZE DISPLAY ARRAY*)
588   4    6:7   200                FILLCHAR(DISPLAY,SIZEOF(DISPLAY),0);
589   4    6:7   209                FOR I := 0 TO 63 DO
590   4    6:8   223                  MOVELEFT(CHARBUF.TRITON[I],DISPLAY[I,10],8);
591   4    6:7   249                UNITWRITE(3,DISPLAY[-80],23)
592   4    6:6   262                END  ELSE
593   4    6:6   264                  UNITWRITE(3,DISPLAY,7);
594   4    6:4   273          END
595   4    6:2   273      END
596   4    6:1   273    ELSE
597   4    6:2   275      SYSCOM^.MISCINFO.HAS8510A := FALSE;
598   4    6:1   284    FCLOSE(LFIB,CNORMAL)
599   4    6:0   288  END (*INITCHARSET*) ;
600   4    6:0   314
601   4    7:0     1  PROCEDURE INITHEAP;
602   4    7:0     1  VAR LWINDOW: WINDOWP;
603   4    7:0     0  BEGIN (*BASIC FILE AND HEAP SETTUP*)
604   4    7:1     0    SYSCOM^.GDIRP := NIL; (* MUST PRECEDE THE FIRST "NEW" EXECUTED *)
605   4    7:1     7    NEW(SWAPFIB,TRUE,FALSE); FINIT(SWAPFIB^,NIL,-1);
606   4    7:1    22    NEW(INPUTFIB,TRUE,FALSE);  NEW(LWINDOW);
607   4    7:1    33    FINIT(INPUTFIB^,LWINDOW,0);
608   4    7:1    41    NEW(OUTPUTFIB,TRUE,FALSE);  NEW(LWINDOW);
609   4    7:1    52    FINIT(OUTPUTFIB^,LWINDOW,0);
610   4    7:1    60    NEW(SYSTERM,TRUE,FALSE);  NEW(LWINDOW);
611   4    7:1    71    FINIT(SYSTERM^,LWINDOW,0);
612   4    7:1    79    GFILES[0] := INPUTFIB; GFILES[1] := OUTPUTFIB;
613   4    7:1    99    WITH USERINFO DO
614   4    7:2    99      BEGIN
615   4    7:3    99        NEW(SYMFIBP,TRUE,FALSE); FINIT(SYMFIBP^,NIL,-1);
616   4    7:3   114        NEW(CODEFIBP,TRUE,FALSE); FINIT(CODEFIBP^,NIL,-1)
617   4    7:2   126      END;
618   4    7:1   129    MARK(EMPTYHEAP)
```

```
619   4     7:0   132   END (*INITHEAP*) ;
620   4     7:0   146
621   4     8:0     1   PROCEDURE INITWORKFILE;
622   4     8:0     1
623   4     9:0     1     PROCEDURE TRY_OPEN(VAR WORK_FIB : FIB; FIRST : FULL_ID; VAR SEC_VOL : VID;
624   4     9:0     4                        VAR SEC_NAME : TID; VAR FLAG : BOOLEAN);
625   4     9:0    18     VAR  LTITLE : FULL_ID;
626   4     9:0     0     BEGIN
627   4     9:1     0       FOPEN(WORK_FIB,FIRST,TRUE,NIL);
628   4     9:1    13       IF NOT WORK_FIB.FISOPEN THEN
629   4     9:2    18         IF SEC_NAME <> '' THEN
630   4     9:3    26           BEGIN
631   4     9:4    26             LTITLE := CONCAT(SEC_VOL,':',SEC_NAME);
632   4     9:4    59             FOPEN(WORK_FIB,LTITLE,TRUE,NIL);
633   4     9:3    67             END;
634   4     9:1    67       FLAG := WORK_FIB.FISOPEN;
635   4     9:1    71       IF FLAG THEN
636   4     9:2    75         BEGIN
637   4     9:3    75           SEC_VOL := WORK_FIB.FVID;
638   4     9:3    81           SEC_NAME := WORK_FIB.FHEADER.DTID
639   4     9:2    83           END;
640   4     9:1    87       FCLOSE(WORK_FIB,CNORMAL);
641   4     9:0    92       END;  [OF TRY_OPEN]
642   4     9:0   104
643   4     8:0     0   BEGIN
644   4     8:1     0     WITH USERINFO DO
645   4     8:2     0       BEGIN (*INITIALIZE WORK FILES ETC*)
646   4     8:3     0         ERRNUM := 0; ERRBLK := 0; ERRSYM := 0;
647   4     8:3    12         IF JUSTBOOTED THEN
648   4     8:4    15           BEGIN
649   4     8:5    15             SYMTID := ''; CODETID := ''; WORKTID := '';
650   4     8:5    39             SYMVID := SYVID; CODEVID := SYVID; WORKVID := SYVID
651   4     8:4    58           END;
652   4     8:3    63         TRY_OPEN(SYMFIBP^,'*SYSTEM.WRK.TEXT',SYMVID,SYMTID,GOTSYM);
653   4     8:3    96         TRY_OPEN(CODEFIBP^,'*SYSTEM.WRK.CODE',CODEVID,CODETID,GOTCODE);
654   4     8:3   129         ALTMODE := SYSCOM^.CRTINFO.ALTMODE;
655   4     8:3   140         SLOWTERM := SYSCOM^.MISCINFO.SLOWTERM;
656   4     8:3   151         STUPID := SYSCOM^.MISCINFO.STUPID;
657   4     8:2   162       END
658   4     8:0   162   END (*INITWORKFILE*) ;
659   4     8:0   174
```

```
660   4   10:0     1 PROCEDURE INITFILES;
661   4   10:0     0   BEGIN
662   4   10:1     0     FCLOSE(SWAPFIB^,CNORMAL);
663   4   10:1     7     FCLOSE(USERINFO.SYMFIBP^,CNORMAL);
664   4   10:1    14     FCLOSE(USERINFO.CODEFIBP^,CNORMAL);
665   4   10:1    21     FCLOSE(INPUTFIB^,CNORMAL);
666   4   10:1    28     FCLOSE(OUTPUTFIB^,CNORMAL);
667   4   10:1    35     LTITLE := 'CONSOLE:';
668   4   10:1    50     FOPEN(INPUTFIB^,LTITLE,TRUE,NIL);
669   4   10:1    60     FOPEN(OUTPUTFIB^,LTITLE,TRUE,NIL);
670   4   10:1    70     IF JUSTBOOTED THEN
671   4   10:2    73       BEGIN LTITLE := 'SYSTERM:';
672   4   10:3    88         FOPEN(SYSTERM^,LTITLE,TRUE,NIL)
673   4   10:2    95       END;
674   4   10:1    98     GFILES[0] := INPUTFIB;
675   4   10:1   108     GFILES[1] := OUTPUTFIB;
676   4   10:1   118     GFILES[2] := SYSTERM;
677   4   10:1   128     GFILES[3] := NIL; GFILES[4] := NIL; GFILES[5] := NIL;
678   4   10:0   152   END (*INITFILES*) ;
679   4   10:0   164
680   4    1:0     0 BEGIN (*INITIALIZE*)
681   4    1:1     0   JUSTBOOTED := EMPTYHEAP = NIL;
682   4    1:1     7   MONTHS[ 0] := '???'; MONTHS[ 1] := 'JAN';
683   4    1:1    33   MONTHS[ 2] := 'FEB'; MONTHS[ 3] := 'MAR';
684   4    1:1    59   MONTHS[ 4] := 'APR'; MONTHS[ 5] := 'MAY';
685   4    1:1    85   MONTHS[ 6] := 'JUN'; MONTHS[ 7] := 'JUL';
686   4    1:1   111   MONTHS[ 8] := 'AUG'; MONTHS[ 9] := 'SEP';
687   4    1:1   137   MONTHS[10] := 'OCT'; MONTHS[11] := 'NOV';
688   4    1:1   163   MONTHS[12] := 'DEC'; MONTHS[13] := '???';
689   4    1:1   189   MONTHS[14] := '???'; MONTHS[15] := '???';
690   4    1:1   215   IF JUSTBOOTED THEN INITHEAP
691   4    1:1   218   ELSE RELEASE(EMPTYHEAP);
692   4    1:1   227   INITUNITABLE; [AND THE DATE, FILENAMES, *SYSTEM.STARTUP]
693   4    1:1   229   INITFILES;
694   4    1:1   231   INITSYSCOM; (*AND SOME GLOBALS*)
695   4    1:1   233   INITWORKFILE;
696   4    1:1   235   CLEARSCREEN;
697   4    1:1   238   IF SYSCOM^.MISCINFO.HAS8510A THEN
698   4    1:2   248     INITCHARSET;
699   4    1:1   250   WRITELN(OUTPUT);
700   4    1:1   256   IF JUSTBOOTED THEN
```

```
701   4    1:2   259     IF NOT STARTUP THEN
702   4    1:3   264       WITH SYSCOM^ DO
703   4    1:4   270         BEGIN
704   4    1:5   270           IF MISCINFO.HASXYCRT THEN
705   4    1:6   280             BEGIN
706   4    1:7   280               FGOTOXY(0,CRTINFO.HEIGHT DIV 3);
707   4    1:7   291               IF FILL_LEN > 0 THEN
708   4    1:8   296                 WRITE(OUTPUT,FILLER);
709   4    1:6   306               END;
710   4    1:5   306           WRITELN(OUTPUT,'WELCOME  ',SYVID,',  TO');
711   4    1:5   356           WRITELN(OUTPUT);
712   4    1:5   362           WRITELN(OUTPUT,'U.C.S.D.  PASCAL  SYSTEM  II.0');
713   4    1:5   408           WRITELN(OUTPUT);
714   4    1:5   414           WITH THEDATE DO
715   4    1:6   414             WRITE(OUTPUT,'CURRENT DATE IS  ',DAY,'-',MONTHS[MONTH],'-',YEAR);
716   4    1:5   500           WRITELN(OUTPUT);
717   4    1:4   506         END  ELSE  [NOTHING]
718   4    1:1   508   ELSE
719   4    1:2   510     WRITELN(OUTPUT,'SYSTEM RE-INITIALIZED')
720   4    1:0   547 END (*INITIALIZE*) ;
721   4    1:0   564
722   4    1:0   564
723   4    1:0   564 [$I SYSSEGS.A  ]
723   4    1:0   564 [$I SYSSEGS.B  ]
724   4    1:0   564
725   4    1:0   564     (*************************************************************)
726   4    1:0   564     (*                                                           *)
727   4    1:0   564     (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.   *)
728   4    1:0   564     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-    *)
729   4    1:0   564     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE   *)
730   4    1:0   564     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
731   4    1:0   564     (*                                                           *)
732   4    1:0   564     (*************************************************************)
733   4    1:0   564
734   4    1:0   564
735   5    1:D     3 SEGMENT FUNCTION GETCMD(LASTST: CMDSTATE): CMDSTATE;
736   5    1:D     4   CONST ASSEMONLY = LINKANDGO;
737   5    1:D     4   TYPE  STATUS_ASSOCIATE = (FOUND_OK,FOUND_BAD,NOT_FOUND);
738   5    1:D     4   VAR CH: CHAR; BADCMD: BOOLEAN;
739   5    1:D     6       DONT_CARE : STATUS_ASSOCIATE;
740   5    1:D     7
```

```
741   5    2:0     1   PROCEDURE RUNWORKFILE(OKTOLINK, RUNONLY: BOOLEAN);
742   5    2:0     3     FORWARD;
743   5    2:0     3
744   5    3:0     3   FUNCTION SYS_ASSOCIATE(SYS_NAME:SYSFILE):BOOLEAN;
745   5    3:0     4     FORWARD;
746   5    3:0     4
747   5    4:0     3   FUNCTION ASSOCIATE(TITLE: STRING; OKTOLINK, RUNONLY,ERROR_OK: BOOLEAN;
748   5    4:0     7                      VAR ASS_STATUS : STATUS_ASSOCIATE): BOOLEAN;
749   5    4:0    49     LABEL 1;
750   5    4:0    49     VAR RSLT: IORSLTWD; LSEG: SEGRANGE;
751   5    4:0    51         SEGTBL: RECORD
752   5    4:0    51                   DISKINFO: ARRAY [SEGRANGE] OF SEGDESC;
753   5    4:0    51                   SEGNAME: ARRAY [SEGRANGE] OF
754   5    4:0    51                              PACKED ARRAY [0..7] OF CHAR;
755   5    4:0    51                   SEGKIND: ARRAY [SEGRANGE] OF
756   5    4:0    51                              (LINKED,HOSTSEG,SEGPROC,UNITSEG,SEPRTSEG);
757   5    4:0    51                   FILLER: ARRAY [0..143] OF INTEGER
758   5    4:0    51                 END [ SEGTBL ] ;
759   5    4:0     0   BEGIN
760   5    4:1     0     ASS_STATUS := NOT_FOUND;
761   5    4:1     8     ASSOCIATE := FALSE;
762   5    4:1    11     FOPEN(USERINFO.CODEFIBP^,TITLE,TRUE,NIL);
763   5    4:1    21     RSLT := SYSCOM^.IORSLT;
764   5    4:1    27     IF RSLT <> INOERROR THEN
765   5    4:2    33       BEGIN
766   5    4:3    33         IF ERROR_OK THEN
767   5    4:4    36           IF RSLT = IBADTITLE THEN
768   5    4:5    42             WRITE(OUTPUT,'ILLEGAL FILE NAME')
769   5    4:4    69           ELSE
770   5    4:5    71             WRITE(OUTPUT,'NO FILE ',TITLE);
771   5    4:3    98         GOTO 1
772   5    4:2   100       END;
773   5    4:1   100     ASS_STATUS := FOUND_BAD; [UNTIL SHOWN OTHERWISE]
774   5    4:1   103     WITH USERINFO,SYSCOM^ DO
775   5    4:2   109       IF CODEFIBP^.FHEADER.DFKIND <> CODEFILE THEN
776   5    4:3   121         BEGIN
777   5    4:4   121           WRITE(OUTPUT,TITLE,' NOT CODE');
778   5    4:4   149           GOTO 1
779   5    4:3   151         END
780   5    4:2   151       ELSE
781   5    4:3   153         BEGIN
```

```
732  5    4:4   153          UNITREAD(CODEFIBP^.FUNIT,SEGTBL,SIZEOF(SEGTBL),
783  5    4:4   163                   CODEFIBP^.FHEADER.DFIRSTBLK);
784  5    4:4   171        IF IORESULT <> ORD(INOERROR) THEN
785  5    4:5   177          BEGIN
786  5    4:6   177            WRITE(OUTPUT,'BAD BLOCK #0');
787  5    4:6   199            GOTO 1
788  5    4:5   201          END;
789  5    4:4   201        WITH SEGTBL DO
790  5    4:5   201          FOR LSEG := 0 TO MAXSEG DO
791  5    4:6   216            IF (SEGKIND[LSEG]<LINKED) OR (SEGKIND[LSEG]>SEPRTSEG) THEN
792  5    4:7   239              BEGIN [ PRE I.5 CODE...FIX UP! ]
793  5    4:8   239                FILLCHAR(SEGKIND, SIZEOF(SEGKIND), ORD(LINKED));
794  5    4:8   247                FILLCHAR(FILLER, SIZEOF(FILLER), 0);
795  5    4:8   257                UNITWRITE(CODEFIBP^.FUNIT, SEGTBL, SIZEOF(SEGTBL),
796  5    4:8   267                          CODEFIBP^.FHEADER.DFIRSTBLK)
797  5    4:7   275              END;
798  5    4:4   283        WITH SEGTBL DO
799  5    4:5   283          FOR LSEG := 0 TO MAXSEG DO
800  5    4:6   298            IF SEGKIND[LSEG] <> LINKED THEN
801  5    4:7   310            BEGIN
802  5    4:8   310              IF OKTOLINK THEN
803  5    4:9   313                BEGIN WRITELN(OUTPUT,'LINKING...');
804  5    4:0   339                  FCLOSE(CODEFIBP^, CNORMAL);
805  5    4:0   346                  IF SYS_ASSOCIATE(LINKER) THEN
806  5    4:1   353                    BEGIN
807  5    4:2   353                      IF RUNONLY THEN GETCMD := LINKANDGO
808  5    4:2   356                      ELSE GETCMD := LINKDEBUG;
809  5    4:2   364                      EXIT(GETCMD)
810  5    4:1   368                    END
811  5    4:9   368                END
812  5    4:8   368              ELSE
813  5    4:9   370                IF NOT (LASTST IN [LINKANDGO, LINKDEBUG]) THEN
814  5    4:0   379                  WRITE(OUTPUT,'MUST L(INK FIRST');
815  5    4:8   405              GOTO 1
816  5    4:7   407            END;
817  5    4:4   415        FOR LSEG := 1 TO MAXSEG DO
818  5    4:5   430          IF (LSEG = 1) OR (LSEG >= 7) THEN
819  5    4:6   441            WITH SEGTABLE[LSEG],SEGTBL.DISKINFO[LSEG] DO
820  5    4:7   462              BEGIN CODEUNIT := CODEFIBP^.FUNIT;
821  5    4:8   470                CODEDESC.CODELENG := CODELENG;
822  5    4:8   480                CODEDESC.DISKADDR := DISKADDR+
```

```
823   5     4:3    469                                                  CODEFIBP^.FHEADER.DFIRSTBLK
824   5     4:7    492                    END
825   5     4:3    496              END;
826   5     4:1    504          ASS_STATUS := FOUND_OK;
827   5     4:1    507          ASSOCIATE := TRUE;
828   5     4:1    510  1:      FCLOSE(USERINFO.CODEFIBP^,CNORMAL)
829   5     4:0    514        END (*ASSOCIATE*) ;
830   5     4:0    540
831   5     3:D      3        FUNCTION SYS_ASSOCIATE[(SYS_NAME:SYSFILE):BOOLEAN];
832   5     3:D      4        VAR  VOL : VID;
833   5     3:D      8             TITLE : TID;
834   5     3:D     16             SEGS : INTEGER;
835   5     3:D     17             KIND : FILEKIND;
836   5     3:D     18             LUNIT : UNITNUM;
837   5     3:D     19             LTITLE : FULL_ID;
838   5     3:D     31             ASS_STATUS : STATUS_ASSOCIATE;
839   5     3:0      0        BEGIN
840   5     3:1      0          SYS_ASSOCIATE := ASSOCIATE(FILENAME[SYS_NAME],FALSE,FALSE,FALSE,ASS_STATUS);
841   5     3:1     18          IF ASS_STATUS = NOT_FOUND THEN
842   5     3:2     24            IF SCANTITLE(FILENAME[SYS_NAME],VOL,TITLE,SEGS,KIND) THEN
843   5     3:3     46              BEGIN
844   5     3:4     46                LUNIT := 0;
845   5     3:4     49                REPEAT
846   5     3:5     49                  LUNIT := LUNIT + 1;
847   5     3:5     55                  WITH UNITABLE[LUNIT] DO
848   5     3:6     64                    IF UISBLKD THEN
849   5     3:7     69                      BEGIN
850   5     3:8     69                        UVID := '';
851   5     3:8     76                        IF FETCHDIR(LUNIT) THEN
852   5     3:9     85                          BEGIN
853   5     3:0     85                            UVID := SYSCOM^.GDIRP^[0].DVID;
854   5     3:0     98                            LTITLE := CONCAT(UVID,':',TITLE);
855   5     3:0    133                            IF LTITLE <> FILENAME[SYS_NAME] THEN
856   5     3:1    146                              IF ASSOCIATE(LTITLE,FALSE,FALSE,FALSE,ASS_STATUS) THEN
857   5     3:2    159                                FILENAME[SYS_NAME] := LTITLE;
858   5     3:9    170                            END;
859   5     3:7    170                        END; [ OF IF ISBLOCKED ...]
860   5     3:4    170                  UNTIL (LUNIT = MAXUNIT) OR (ASS_STATUS IN [FOUND_OK,FOUND_BAD]);
861   5     3:4    182                SYS_ASSOCIATE := ASS_STATUS = FOUND_OK;
862   5     3:4    188                IF ASS_STATUS = NOT_FOUND THEN
863   5     3:5    194                  IF ASSOCIATE(FILENAME[SYS_NAME],FALSE,FALSE,TRUE,ASS_STATUS) THEN;
```

```
 864   5     3:5   212                          {JUST TO GET THE APPROPRIATE ERROR}
 865   5     3:3   212            END; {OF IF SCANTITLE...}
 866   5     3:0   212      END;  {OF SYS_ASSOCIATE}
 867   5     3:0   230
 868   5     5:D     1 PROCEDURE STARTCOMPILE(NEXTST: CMDSTATE);
 869   5     5:D     2     LABEL 1;
 870   5     5:D     2     VAR TEXT_TITLE,TITLE: STRING[40];
 871   5     5:D    44         I : INTEGER;
 872   5     5:D    45         CODE_NAME : FULL_ID;
 873   5     5:D    57         SYS_TYPE : SYSFILE;
 874   5     5:0     0   BEGIN
 875   5     5:1     0     IF NEXTST = ASSEMONLY THEN
 876   5     5:2     5       WRITE(OUTPUT,'ASSEMBLING')
 877   5     5:1    25     ELSE
 878   5     5:2    27       WRITE(OUTPUT,'COMPILING');
 879   5     5:1    46     WRITELN(OUTPUT,'...');
 880   5     5:1    65     IF NEXTST = ASSEMONLY THEN
 881   5     5:2    70       SYS_TYPE := ASSMBLER
 882   5     5:1    70     ELSE
 883   5     5:2    75       SYS_TYPE := COMPILER;
 884   5     5:1    78     IF SYS_ASSOCIATE(SYS_TYPE) THEN
 885   5     5:2    86       WITH USERINFO DO
 886   5     5:3    86         BEGIN
 887   5     5:4    86           IF GOTSYM THEN
 888   5     5:5    91             TITLE := CONCAT(SYMVID,':',SYMTID)
 889   5     5:4   126           ELSE
 890   5     5:5   130             BEGIN
 891   5     5:6   130               IF NEXTST = ASSEMONLY THEN
 892   5     5:7   135                 WRITE(OUTPUT, 'ASSEMBLE')
 893   5     5:6   153               ELSE
 894   5     5:7   155                 WRITE(OUTPUT, 'COMPILE');
 895   5     5:6   172               WRITE(OUTPUT,' WHAT TEXT? ');
 896   5     5:6   194               READLN(INPUT, TEXT_TITLE);
 897   5     5:6   209               IF TEXT_TITLE = '' THEN GOTO 1;
 898   5     5:6   220               TITLE := CONCAT(TEXT_TITLE,'.TEXT');
 899   5     5:5   251             END;
 900   5     5:4   251           FOPEN(SYMFIBP^,TITLE,TRUE,NIL);
 901   5     5:4   261           IF IORESULT <> ORD(INOERROR) THEN
 902   5     5:5   267             BEGIN
 903   5     5:6   267               WRITE(OUTPUT,'CAN''T FIND ', TITLE);
 904   5     5:6   297               GOTSYM := FALSE; GOTO 1
```

```
905   5     5:5    303            END;
906   5     5:4    303          TITLE := CONCAT(COPY(FILENAME[SYS_TYPE],1,
907   5     5:4    322                          POS(':',FILENAME[SYS_TYPE])),'SYSTEM.SWAPDISK');
908   5     5:4    377          FOPEN(SWAPFIB^,TITLE,TRUE,NIL);
909   5     5:4    387          CODE_NAME := '*SYSTEM.WRK.CODE[*]';
910   5     5:4    413          IF NOT GOTSYM THEN
911   5     5:5    419            BEGIN
912   5     5:6    419              WRITE(OUTPUT, 'TO WHAT CODEFILE? ');
913   5     5:6    447              READLN(INPUT, TITLE);
914   5     5:6    462              IF TITLE <> '' THEN
915   5     5:7    471                IF TITLE[1] = SYSCOM^.CRTINFO.ALTMODE THEN
916   5     5:8    486                  GOTO 1  ELSE
917   5     5:8    490                  BEGIN       [TREAT '$' AS A WILDCARD]
918   5     5:9    490                    I := POS('$',TITLE);
919   5     5:9    503                    WHILE I <> 0 DO
920   5     5:0    509                      BEGIN
921   5     5:1    509                        DELETE(TITLE,I,1);
922   5     5:1    517                        INSERT(COPY(TEXT_TITLE,1,LENGTH(TEXT_TITLE)),
923   5     5:1    531                               TITLE,I);
924   5     5:1    539                        I := POS('$',TITLE);
925   5     5:0    552                        END;
926   5     5:9    554                    IF TITLE[LENGTH(TITLE)] <> ']' THEN
927   5     5:0    565                      CODE_NAME := CONCAT(TITLE,'.CODE[*]')  ELSE
928   5     5:0    601                      CODE_NAME := TITLE;
929   5     5:8    607                    END;
930   5     5:5    607              END;
931   5     5:4    607            FOPEN(CODEFIBP^,CODE_NAME,FALSE,NIL);
932   5     5:4    617            IF IORESULT <> ORD(INOERROR) THEN
933   5     5:5    623              BEGIN
934   5     5:6    623                WRITE(OUTPUT,'CAN''T OPEN ',CODE_NAME);
935   5     5:6    653                GOTO 1
936   5     5:5    655              END;
937   5     5:4    655            ERRNUM := 0; ERRBLK := 0; ERRSYM := 0;
938   5     5:4    667            IF NEXTST = ASSEMONLY THEN
939   5     5:5    672              NEXTST := COMPONLY;
940   5     5:4    675            GETCMD := NEXTST; EXIT(GETCMD);
941   5     5:4    682          1:
942   5     5:4    682            FCLOSE(SYMFIBP^,CNORMAL);
943   5     5:4    689            FCLOSE(SWAPFIB^,CNORMAL);
944   5     5:3    696          END;
945   5     5:0    696    END (*STARTCOMPILE*) ;
```

```
946   5     5:0   718
947   5     6:D     1   PROCEDURE FINISHCOMPILE;
948   5     6:D     1   VAR  RESULT : INTEGER;
949   5     6:0     0   BEGIN
950   5     6:1     0     FCLOSE(USERINFO.SYMFIBP^,CNORMAL);
951   5     6:1     7     FCLOSE(SWAPFIB^,CNORMAL);
952   5     6:1    14     IF SYSCOM^.MISCINFO.HAS8510A THEN
953   5     6:2    24       UNITCLEAR(3);
954   5     6:1    27     WITH USERINFO DO
955   5     6:2    27       IF ERRNUM > 0 THEN
956   5     6:3    34         BEGIN GOTCODE := FALSE;
957   5     6:4    38           FCLOSE(CODEFIBP^,CPURGE);
958   5     6:4    45           IF ERRBLK > 0 THEN
959   5     6:5    52             BEGIN CLEARSCREEN; WRITELN(OUTPUT);
960   5     6:6    61               IF SYS_ASSOCIATE(EDITOR) THEN
961   5     6:7    68                 BEGIN GETCMD := SYSPROG; EXIT(GETCMD) END
962   5     6:5    75             END
963   5     6:3    75         END
964   5     6:2    75       ELSE
965   5     6:3    77         BEGIN
966   5     6:4    77           IF CODETID <> 'SYSTEM.WRK.CODE' THEN
967   5     6:5   102             BEGIN
968   5     6:6   102               CODEVID := CODEFIBP^.FVID;
969   5     6:6   112               CODETID := CODEFIBP^.FHEADER.DTID;
970   5     6:6   122               IF CODETID <> 'SYSTEM.WRK.CODE' THEN
971   5     6:7   147                 BEGIN
972   5     6:8   147                   WORKVID := CODEVID;
973   5     6:8   155                   IF LENGTH(CODETID) > 5 THEN
974   5     6:9   164                     IF COPY(CODETID,LENGTH(CODETID)-4,5) = '.CODE' THEN
975   5     6:0   194                       WORKTID := COPY(CODETID,1,LENGTH(CODETID)-5);
976   5     6:7   217                   END;
977   5     6:5   217                 END;
978   5     6:4   217             GOTCODE := TRUE;
979   5     6:4   221             [FIB FOR CODEFILE WAS CLOSED IN COMMAND]
980   5     6:4   221             IF LASTST IN [COMPANDGO,COMPDEBUG] THEN
981   5     6:5   229               RUNWORKFILE(TRUE, LASTST = COMPANDGO)
982   5     6:3   233         END
983   5     6:0   235   END (*FINISHCOMPILE*) ;
984   5     6:0   250
985   5     7:D     1   PROCEDURE EXECUTE;
986   5     7:D     1     VAR TITLE: STRING[255];
```

```
 987    5     7:0      0   BEGIN
 988    5     7:1      0     WRITE(OUTPUT,'EXECUTE');
 989    5     7:1     17     IF NOT SYSCOM^.MISCINFO.SLOWTERM THEN
 990    5     7:2     28       WRITE(OUTPUT,' WHAT FILE');
 991    5     7:1     48     WRITE(OUTPUT,'? '); READLN(TITLE);
 992    5     7:1     77     IF LENGTH(TITLE) > 0 THEN
 993    5     7:2     85       BEGIN
 994    5     7:3     85         IF TITLE[LENGTH(TITLE)] = '.' THEN
 995    5     7:4     96           DELETE(TITLE,LENGTH(TITLE),1)
 996    5     7:3    106         ELSE
 997    5     7:4    108           INSERT('.CODE',TITLE,LENGTH(TITLE)+1);
 998    5     7:3    130         IF ASSOCIATE(TITLE, FALSE, FALSE, TRUE, DONT_CARE) THEN
 999    5     7:4    143           BEGIN GETCMD := SYSPROG; EXIT(GETCMD) END
1000    5     7:2    150       END
1001    5     7:0    150   END (*EXECUTE*) ;
1002    5     7:0    162
1003    5     2:D      1   PROCEDURE RUNWORKFILE;
1004    5     2:0      0   BEGIN
1005    5     2:1      0     WITH USERINFO DO
1006    5     2:2      0     IF GOTCODE THEN
1007    5     2:3      5       BEGIN
1008    5     2:4      5         CLEARSCREEN;
1009    5     2:4      8         WRITELN(OUTPUT);
1010    5     2:4     14         IF ASSOCIATE(CONCAT(CODEVID,':',CODETID), OKTOLINK, RUNONLY, TRUE,
1011    5     2:4     50                      DONT_CARE) THEN
1012    5     2:5     58           BEGIN
1013    5     2:6     58             WRITELN(OUTPUT,'RUNNING...');
1014    5     2:6     84             IF RUNONLY THEN
1015    5     2:7     87                 GETCMD := SYSPROG
1016    5     2:6     87             ELSE
1017    5     2:7     92                 GETCMD := DEBUGCALL;
1018    5     2:6     95             EXIT(GETCMD)
1019    5     2:5     99           END;
1020    5     2:4     99         IF NOT (LASTST IN [LINKANDGO, LINKDEBUG]) THEN
1021    5     2:5    108           GOTCODE := FALSE
1022    5     2:3    108       END
1023    5     2:2    112     ELSE
1024    5     2:3    114       IF RUNONLY THEN
1025    5     2:4    117         STARTCOMPILE(COMPANDGO)
1026    5     2:3    118       ELSE
1027    5     2:4    122         STARTCOMPILE(COMPDEBUG)
```

```
1028   5    2:0   123   END { RUNWORKFILE } ;
1029   5    2:0   133
1030   5    1:0     0 BEGIN (*GETCMD*)
1031   5    1:1     0   INPUTFIB^.FEOF := FALSE;
1032   5    1:1     7   OUTPUTFIB^.FEOF := FALSE;
1033   5    1:1    14   SYSTERM^.FEOF := FALSE;
1034   5    1:1    21   GFILES[0] := INPUTFIB; GFILES[1] := OUTPUTFIB;
1035   5    1:1    41   IF LASTST = HALTINIT THEN
1036   5    1:2    46     IF ASSOCIATE('*SYSTEM.STARTUP',FALSE,FALSE,FALSE,DONT_CARE) THEN
1037   5    1:3    75       BEGIN CLEARSCREEN;
1038   5    1:4    78         GETCMD := SYSPROG; EXIT(GETCMD)
1039   5    1:3    85       END;
1040   5    1:1    85   IF LASTST IN [COMPONLY,COMPANDGO,COMPDEBUG] THEN
1041   5    1:2    93     FINISHCOMPILE;
1042   5    1:1    95   IF LASTST IN [LINKANDGO,LINKDEBUG] THEN
1043   5    1:2   103     RUNWORKFILE(FALSE, LASTST = LINKANDGO);
1044   5    1:1   109   IF SYSCOM^.MISCINFO.USERKIND = AQUIZ THEN
1045   5    1:2   121     IF LASTST = HALTINIT THEN
1046   5    1:3   126       BEGIN LASTST := COMPANDGO; RUNWORKFILE(TRUE, TRUE) END
1047   5    1:2   133     ELSE
1048   5    1:3   135       BEGIN
1049   5    1:4   135         EMPTYHEAP := NIL;
1050   5    1:4   139         GETCMD := HALTINIT;
1051   5    1:4   142         EXIT(GETCMD)
1052   5    1:3   146       END;
1053   5    1:1   146   WITH USERINFO DO
1054   5    1:2   146     BEGIN ERRNUM := 0; ERRBLK := 0; ERRSYM := 0 END;
1055   5    1:1   158   BADCMD := FALSE;
1056   5    1:1   161   REPEAT
1057   5    1:2   161     PL :=
1058   5    1:2   164 'COMMAND: E(DIT, R(UN, F(ILE, C(OMP, L(INK, X(ECUTE, A(SSEM, D(EBUG,? [II.0]';
1059   5    1:2   244     PROMPT; CH := GETCHAR(BADCMD); CLEARSCREEN;
1060   5    1:2   258     IF CH = '?' THEN
1061   5    1:3   263       BEGIN PL := 'COMMAND: U(SER RESTART, I(NITIALIZE, H(ALT';
1062   5    1:4   313         PROMPT; CH := GETCHAR(BADCMD); CLEARSCREEN
1063   5    1:3   324       END;
1064   5    1:2   327     BADCMD := NOT (CH IN ['E','R','F','C','L','X','A','D','U','I','H','?']);
1065   5    1:2   347     IF NOT BADCMD THEN
1066   5    1:3   351       CASE CH OF
1067   5    1:3   354         'E':  BEGIN WRITELN(OUTPUT);
1068   5    1:5   360                 IF SYS_ASSOCI( '(EDITOR) THEN
```

```
1069  5   1:6   367                    BEGIN GETCMD := SYSPROG; EXIT(GETCMD) END
1070  5   1:4   374                  END;
1071  5   1:3   376          'F':  BEGIN WRITELN(OUTPUT);
1072  5   1:5   382                  IF SYS_ASSOCIATE(FILER) THEN
1073  5   1:6   389                    BEGIN GETCMD := SYSPROG; EXIT(GETCMD) END
1074  5   1:4   396                  END;
1075  5   1:3   398          'L':  BEGIN WRITELN(OUTPUT,'LINKING...');
1076  5   1:5   424                  IF SYS_ASSOCIATE(LINKER) THEN
1077  5   1:6   431                    BEGIN GETCMD := SYSPROG; EXIT(GETCMD) END
1078  5   1:4   438                  END;
1079  5   1:3   440          'X':  EXECUTE;
1080  5   1:3   444          'C':  STARTCOMPILE(COMPONLY);
1081  5   1:3   449          'A':  STARTCOMPILE(ASSEMONLY);
1082  5   1:3   454          'U':  IF LASTST <> UPROGNOU THEN
1083  5   1:5   459                  BEGIN
1084  5   1:6   459                    WRITELN(OUTPUT,'RESTARTING...');
1085  5   1:6   488                    GETCMD := SYSPROG; EXIT(GETCMD)
1086  5   1:5   495                  END
1087  5   1:4   495                ELSE
1088  5   1:5   497                  BEGIN WRITELN(OUTPUT); WRITE(OUTPUT,'U NOT ALLOWED') END;
1089  5   1:3   528      'R','D':  RUNWORKFILE(TRUE, CH = 'R');
1090  5   1:3   536      'I','H':  BEGIN
1091  5   1:5   536                  GETCMD := HALTINIT;
1092  5   1:5   539                  IF CH = 'H' THEN
1093  5   1:6   544                    EMPTYHEAP := NIL;
1094  5   1:5   548                  EXIT(GETCMD)
1095  5   1:4   552                END
1096  5   1:3   552        END
1097  5   1:1   610    UNTIL FALSE
1098  5   1:0   610 END (*GETCMD*) ;
1099  5   1:0   634 [$I SYSSEGS.B  ]
1099  5   1:0   634 [$I SYSTEM.A ]
1100  5   1:0   634
1101  5   1:0   634      (*****************************************************************)
1102  5   1:0   634      (*                                                               *)
1103  5   1:0   634      (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
1104  5   1:0   634      (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
1105  5   1:0   634      (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
1106  5   1:0   634      (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
1107  5   1:0   634      (*                                                               *)
1108  5   1:0   634      (*****************************************************************)
```

```
1109    5     1:0    634
1110    5     1:0    634
1111    0     2:0      1 PROCEDURE EXECERROR;
1112    0     2:0      1
1113    0    44:0      1   PROCEDURE PRINTLOCS;
1114    0    44:0      0   BEGIN
1115    0    44:1      0     WITH SYSCOM^,SYSCOM^.BOMBP^ DO
1116    0    44:2     11       BEGIN
1117    0    44:3     11         WRITE(OUTPUT,'S# ',MSSEG^.BYTE[0],
1118    0    44:3     35                      ', P# ',MSJTAB^.BYTE[0],
1119    0    44:3     61                      ', I# ');
1120    0    44:3     76         IF MISCINFO.IS_FLIPT THEN
1121    0    44:4     84           WRITELN(MSIPC)
1122    0    44:3     99         ELSE
1123    0    44:4    101           WRITELN(MSIPC - (ORD(MSJTAB) - 2 - MSJTAB^.WORD[-1]));
1124    0    44:2    129       END;
1125    0    44:0    129     END [OF PRINTLOCS];
1126    0    44:0    142
1127    0     2:0      0 BEGIN
1128    0     2:1      0   WITH SYSCOM^ DO
1129    0     2:2      5     BEGIN
1130    0     2:3      5       IF XEQERR = 4 THEN
1131    0     2:4     11         BEGIN RELEASE(EMPTYHEAP);
1132    0     2:5     16           PL := '*STK OFLOW*';
1133    0     2:5     35           UNITWRITE(2,PL[1],LENGTH(PL));
1134    0     2:5     49           EXIT(COMMAND)
1135    0     2:4     53         END;
1136    0     2:3     53       BOMBP^.MSIPC := BOMBIPC;
1137    0     2:3     61       IF BUGSTATE <> 0 THEN
1138    0     2:4     67         BEGIN DEBUGGER; XEQERR := 0 END
1139    0     2:3     75       ELSE
1140    0     2:4     77         BEGIN RELEASE(EMPTYHEAP);
1141    0     2:5     82           GFILES[0] := INPUTFIB; GFILES[1] := OUTPUTFIB;
1142    0     2:5    102           BOMBIPC := IORESULT; FWRITELN(SYSTERM^);
1143    0     2:5    113           IF SYSUNIT = VOLSEARCH(SYVID,FALSE,SYSCOM^.GDIRP[WATCH OUT]) THEN
1144    0     2:6    131             PRINTERROR(XEQERR,BOMBIPC)
1145    0     2:5    136           ELSE
1146    0     2:6    141             BEGIN
1147    0     2:7    141               WRITELN(OUTPUT,'EXEC ERR # ',XEQERR);
1148    0     2:7    177               IF XEQERR = 10 THEN
1149    0     2:8    183                 WRITE(OUTPUT,'  ',BOMBIPC)
```

```
1150   0     2:6   201              END;
1151   0     2:5   201            PRINTLOCS;
1152   0     2:5   203            IF NOT SPACEWAIT(TRUE) THEN EXIT(COMMAND)
1153   0     2:4   215          END
1154   0     2:2   215      END
1155   0     2:0   215 END (*EXECERROR*) ;
1156   0     2:0   230
1157   0    45:0     3 FUNCTION CHECKDEL(CH: CHAR; VAR SINX: INTEGER): BOOLEAN;
1158   0    45:0     0 BEGIN CHECKDEL := FALSE;
1159   0    45:1     3   WITH SYSCOM^,CRTCTRL DO
1160   0    45:2    13     BEGIN
1161   0    45:3    13       IF CH = CRTINFO.LINEDEL THEN
1162   0    45:4    23         BEGIN CHECKDEL := TRUE;
1163   0    45:5    26           IF (BACKSPACE = CHR(0)) OR (ERASEEOL = CHR(0)) THEN
1164   0    45:6    45             BEGIN SINX := 1;
1165   0    45:7    48               WRITELN(OUTPUT,'<DEL')
1166   0    45:6    68             END
1167   0    45:5    68           ELSE
1168   0    45:6    70             BEGIN
1169   0    45:7    70               WHILE SINX > 1 DO
1170   0    45:8    76                 BEGIN SINX := SINX-1; WRITE(OUTPUT,BACKSPACE) END;
1171   0    45:7    97               WRITE(OUTPUT,ESCAPE,ERASEEOL)
1172   0    45:6   121             END
1173   0    45:4   121         END;
1174   0    45:3   121       IF CH = CRTINFO.CHARDEL THEN
1175   0    45:4   131         BEGIN CHECKDEL := TRUE;
1176   0    45:5   134           IF SINX > 1 THEN
1177   0    45:6   140             BEGIN SINX := SINX-1;
1178   0    45:7   146               IF BACKSPACE = CHR(0) THEN
1179   0    45:8   156                 IF CRTINFO.CHARDEL < ' ' THEN
1180   0    45:9   166                   WRITE(OUTPUT,'_')
1181   0    45:8   174                 ELSE (*ASSUME PRINTABLE*)
1182   0    45:7   176               ELSE
1183   0    45:8   178                 BEGIN
1184   0    45:9   178                   IF CRTINFO.CHARDEL <> BACKSPACE THEN
1185   0    45:0   193                     WRITE(OUTPUT,BACKSPACE);
1186   0    45:9   206                   WRITE(OUTPUT,' ',BACKSPACE)
1187   0    45:8   227                 END
1188   0    45:6   227             END
1189   0    45:5   227           ELSE
1190   0    45:6   229             IF CRTINFO.CHARDEL = BACKSPACE THEN
```

```
1191    0    45:7    244                WRITE(OUTPUT,' ')
1192    0    45:4    252            END
1193    0    45:2    252        END
1194    0    45:0    252 END (*CHECKDEL*) ;
1195    0    45:0    266
1196    0    45:0    266
1197    0    46:D      1 PROCEDURE PUTPREFIXED(WHICH:INTEGER; COMMANDCHAR:CHAR);
1198    0    46:0      0 BEGIN
1199    0    46:1      0   WITH SYSCOM^ DO
1200    0    46:2      5     IF COMMANDCHAR <> CHR(0) THEN
1201    0    46:3     10       BEGIN
1202    0    46:4     10         IF CRTCTRL.PREFIXED[WHICH] THEN
1203    0    46:5     20           WRITE(OUTPUT,CRTCTRL.ESCAPE);
1204    0    46:4     33         WRITE(OUTPUT,COMMANDCHAR);
1205    0    46:4     41         IF FILL_LEN>0 THEN
1206    0    46:5     46           WRITE(OUTPUT,FILLER);
1207    0    46:3     56       END;
1208    0    46:0     56 END;
1209    0    46:0     68
1210    0    36:D      1 PROCEDURE HOMECURSOR;
1211    0    36:0      0 BEGIN
1212    0    36:1      0   PUTPREFIXED(4,SYSCOM^.CRTCTRL.HOME);
1213    0    36:0     11 END (*HOMECURSOR*) ;
1214    0    36:0     24
1215    0    37:D      1 PROCEDURE CLEARSCREEN;
1216    0    37:0      0 BEGIN HOMECURSOR;
1217    0    37:1      2   WITH SYSCOM^,CRTCTRL DO
1218    0    37:2     12     BEGIN
1219    0    37:3     12       UNITCLEAR(3);
1220    0    37:3     15       IF ERASEEOS <> CHR(0) THEN
1221    0    37:4     25         PUTPREFIXED(3,ERASEEOS)
1222    0    37:3     32       ELSE
1223    0    37:4     36         PUTPREFIXED(6,CLEARSCREEN)
1224    0    37:2     43     END
1225    0    37:0     45 END (*CLEARSCREEN*) ;
1226    0    37:0     58
1227    0    38:D      1 PROCEDURE CLEARLINE;
1228    0    38:0      0 BEGIN
1229    0    38:1      0   PUTPREFIXED(2,SYSCOM^.CRTCTRL.ERASEEOL)
1230    0    38:0      9 END (*CLEARLINE*) ;
1231    0    38:0     24
```

```
1232   0   39:0     1 PROCEDURE PROMPT;
1233   0   39:0     1   VAR I: INTEGER;
1234   0   39:0     0 BEGIN HOMECURSOR;
1235   0   39:1     2   WITH SYSCOM^ DO
1236   0   39:2     7     BEGIN
1237   0   39:3     7       CLEARLINE;
1238   0   39:3     9       IF MISCINFO.SLOWTERM THEN
1239   0   39:4    17         BEGIN
1240   0   39:5    17           I := SCAN(LENGTH(PL),=':',PL[1]);
1241   0   39:5    33           IF I <> LENGTH(PL) THEN PL[0] := CHR(I+1)
1242   0   39:4    49         END
1243   0   39:2    50     END;
1244   0   39:1    50   WRITE(OUTPUT,PL)
1245   0   39:0    60 END (*PROMPT*) ;
1246   0   39:0    72
1247   0   29:0     1 PROCEDURE FGOTOXY(*X,Y: INTEGER*);
1248   0   29:0     0 BEGIN (*ASSUME DATA MEDIA*)
1249   0   29:1     0   WITH SYSCOM^.CRTINFO DO
1250   0   29:2     7     BEGIN
1251   0   29:3     7       IF X < 0 THEN X := 0;
1252   0   29:3    15       IF X > WIDTH THEN X := WIDTH;
1253   0   29:3    25       IF Y < 0 THEN Y := 0;
1254   0   29:3    33       IF Y > HEIGHT THEN Y := HEIGHT
1255   0   29:2    39     END;
1256   0   29:1    43   WRITE(OUTPUT,CHR(30),CHR(X+32),CHR(Y+32))
1257   0   29:0    71 END (*GOTOXY*) ;
1258   0   29:0    84
1259   0   41:0     3 FUNCTION GETCHAR(*FLUSH: BOOLEAN*);
1260   0   41:0     4   VAR CH: CHAR;
1261   0   41:0     0 BEGIN
1262   0   41:1     0   IF FLUSH THEN UNITCLEAR(1);
1263   0   41:1     6   IF INPUTFIB^.FEOF THEN EXIT(COMMAND);
1264   0   41:1    16   INPUTFIB^.FSTATE := FNEEDCHAR;
1265   0   41:1    23   READ(INPUT,CH);
1266   0   41:1    31   IF (CH >= 'A') AND (CH <= 'Z') THEN
1267   0   41:2    40     CH := CHR(ORD(CH)-ORD('A')+ORD('A'));
1268   0   41:1    47   GETCHAR := CH
1269   0   41:0    47 END (*GETCHAR*) ;
1270   0   41:0    62
1271   0   40:0     3 FUNCTION SPACEWAIT(*FLUSH: BOOLEAN*);
1272   0   40:0     4   VAR CH: CHAR;
```

```
1273   0    40:0      0 BEGIN
1274   0    40:1      0   REPEAT
1275   0    40:2      0     WRITE(OUTPUT,'TYPE <SPACE>');
1276   0    40:2     22     IF NOT SYSCOM^.MISCINFO.SLOWTERM THEN
1277   0    40:3     33       WRITE(OUTPUT,' TO CONTINUE');
1278   0    40:2     55     CH := GETCHAR(FLUSH);
1279   0    40:2     62     IF NOT EOLN(INPUT) THEN
1280   0    40:3     73       WRITELN(OUTPUT);
1281   0    40:2     79     CLEARLINE
1282   0    40:1     79   UNTIL (CH = ' ') OR (CH = SYSCOM^.CRTINFO.ALTMODE);
1283   0    40:1     97   SPACEWAIT := CH <> ' '
1284   0    40:0     98 END (*SPACEWAIT*) ;
1285   0    40:0    116
1286   0     1:D      3 FUNCTION SCANTITLE(*FTITLE: STRING; VAR FVID: VID; VAR FTID: TID;
1287   0    33:D     49                          VAR FSEGS: INTEGER; VAR FKIND: FILEKIND*);
1288   0    33:D     49   VAR I,RBRACK: INTEGER; CH: CHAR; OK: BOOLEAN;
1289   0    33:0      0 BEGIN
1290   0    33:1      0   FVID := ''; FTID := '';
1291   0    33:1     17   FSEGS := 0; FKIND := UNTYPEDFILE;
1292   0    33:1     23   SCANTITLE := FALSE; I := 1;
1293   0    33:1     29   WHILE I <= LENGTH(FTITLE) DO
1294   0    33:2     38     BEGIN CH := FTITLE[I];
1295   0    33:3     45       IF CH <= ' ' THEN DELETE(FTITLE,I,1)
1296   0    33:3     59       ELSE
1297   0    33:4     61         BEGIN
1298   0    33:5     61           IF (CH >= 'A') AND (CH <= 'Z') THEN
1299   0    33:6     72             FTITLE[I] := CHR(ORD(CH)-ORD('A')+ORD('A'));
1300   0    33:5     83           I := I+1
1301   0    33:4     85         END
1302   0    33:2     89     END;
1303   0    33:1     91   IF LENGTH(FTITLE) > 0 THEN
1304   0    33:2     99     BEGIN
1305   0    33:3     99       IF FTITLE[1] = '*' THEN
1306   0    33:4    107         BEGIN FVID := SYVID; DELETE(FTITLE,1,1) END;
1307   0    33:3    120       I := POS(':',FTITLE);
1308   0    33:3    133       IF I <= 1 THEN
1309   0    33:4    139         BEGIN
1310   0    33:5    139           IF LENGTH(FVID) = 0 THEN FVID := DKVID;
1311   0    33:5    152           IF I = 1 THEN DELETE(FTITLE,1,1)
1312   0    33:4    165         END
1313   0    33:3    165       ELSE
```

```
1314   0    33:4    167          IF I-1 <= VIDLENG THEN
1315   0    33:5    175            BEGIN
1316   0    33:6    175              FVID := COPY(FTITLE,1,I-1);
1317   0    33:6    192              DELETE(FTITLE,1,I)
1318   0    33:5    200            END;
1319   0    33:3    200        IF LENGTH(FVID) > 0 THEN
1320   0    33:4    207          BEGIN
1321   0    33:5    207            I := POS('[',FTITLE);
1322   0    33:5    220            IF I > 0 THEN I := I-1
1323   0    33:5    228            ELSE I := LENGTH(FTITLE);
1324   0    33:5    240            IF I <= TIDLENG THEN
1325   0    33:6    246              BEGIN
1326   0    33:7    246                IF I > 0 THEN
1327   0    33:8    252                  BEGIN FTID := COPY(FTITLE,1,I); DELETE(FTITLE,1,I) END;
1328   0    33:7    275                IF LENGTH(FTITLE) = 0 THEN OK := TRUE
1329   0    33:7    283                ELSE
1330   0    33:8    288                  BEGIN OK := FALSE;
1331   0    33:9    291                    RBRACK := POS(']',FTITLE);
1332   0    33:9    304                    IF RBRACK = 2 THEN OK := TRUE
1333   0    33:9    310                    ELSE
1334   0    33:0    315                      IF RBRACK > 2 THEN
1335   0    33:1    321                        BEGIN OK := TRUE; I := 2;
1336   0    33:2    327                          REPEAT CH := FTITLE[I];
1337   0    33:3    334                            IF CH IN DIGITS THEN
1338   0    33:4    345                              FSEGS := FSEGS*10+(ORD(CH)-ORD('0'))
1339   0    33:3    354                            ELSE OK := FALSE;
1340   0    33:3    361                            I := I+1
1341   0    33:2    363                          UNTIL (I = RBRACK) OR NOT OK;
1342   0    33:2    378                          IF (I = 3) AND (RBRACK = 3) THEN
1343   0    33:3    389                            IF FTITLE[I-1] = '*' THEN
1344   0    33:4    400                              BEGIN FSEGS := -1; OK := TRUE END
1345   0    33:1    407                        END
1346   0    33:8    407                  END;
1347   0    33:7    407                SCANTITLE := OK;
1348   0    33:7    411                IF OK AND (LENGTH(FTID) > 5) THEN
1349   0    33:8    421                  BEGIN
1350   0    33:9    421                    FTITLE := COPY(FTID,LENGTH(FTID)-4,5);
1351   0    33:9    439                    IF FTITLE = '.TEXT' THEN FKIND := TEXTFILE
1352   0    33:9    454                    ELSE
1353   0    33:0    458                    IF FTITLE = '.CODE' THEN FKIND := CODEFILE
1354   0    33:0    473                    ELSE
```

```
1355   0   33:1   477                    IF FTITLE = '.BACK' THEN FKIND := TEXTFILE
1356   0   33:1   492                    ELSE
1357   0   33:2   496                    IF FTITLE = '.INFO' THEN FKIND := INFOFILE
1358   0   33:2   511                    ELSE
1359   0   33:3   515                    IF FTITLE = '.GRAF' THEN FKIND := GRAFFILE
1360   0   33:3   530                    ELSE
1361   0   33:4   534                    IF FTITLE = '.FOTO' THEN FKIND := FOTOFILE
1362   0   33:8   549                  END
1363   0   33:6   551                END
1364   0   33:4   551              END
1365   0   33:2   551          END
1366   0   33:0   551 END (*SCANTITLE*) ;
1367   0   33:0   576
1368   0   33:0   576 (* VOLUME AND DIRECTORY HANDLERS *)
1369   0   33:0   576
1370   0   42:D     3 FUNCTION FETCHDIRC(FUNIT: UNITNUM): BOOLEAN];
1371   0   42:D     4   VAR LINX: DIRRANGE; OK: BOOLEAN; HNOW: INTEGER;
1372   0   42:0     0 BEGIN FETCHDIR := FALSE;
1373   0   42:1     3   WITH SYSCOM^,UNITABLEC[FUNIT] DO
1374   0   42:2    16     BEGIN (*READ IN AND VALIDATE DIR*)
1375   0   42:3    16       IF GDIRP = NIL THEN NEW(GDIRP);
1376   0   42:3    30       UNITREAD(FUNIT,GDIRP^,SIZEOF(DIRECTORY),DIRBLK);
1377   0   42:3    41       OK := IORSLT = INOERROR;
1378   0   42:3    47       IF OK THEN
1379   0   42:4    50         WITH GDIRP^[0] DO
1380   0   42:5    57           BEGIN OK := FALSE; (*CHECK OUT DIR*)
1381   0   42:6    60             IF (DFIRSTBLK = 0) AND
1382   0   42:6    64                 (  (MISCINFO.USERKIND=BOOKER)
1383   0   42:6    72               OR ( (MISCINFO.USERKIND IN [AQUIZ,PQUIZ]) AND (DFKIND=SECUREDIR) )
1384   0   42:6    90                   OR ( (MISCINFO.USERKIND=NORMAL) AND (DFKIND=UNTYPEDFILE) ) )
1385   0   42:6   109               THEN
1386   0   42:7   112               IF (LENGTH(DVID) > 0) AND (LENGTH(DVID) <= VIDLENG) AND
1387   0   42:7   127                  (DNUMFILES >= 0) AND (DNUMFILES <= MAXDIR) THEN
1388   0   42:8   141                 BEGIN OK := TRUE; (*SO FAR SO GOOD*)
1389   0   42:9   144                   IF DVID <> UVID THEN
1390   0   42:0   152                     BEGIN (*NEW VOLUME IN UNIT...CAREFUL*)
1391   0   42:1   152                       LINX := 1;
1392   0   42:1   155                       WHILE LINX <= DNUMFILES DO
1393   0   42:2   162                         WITH GDIRP^[LINX] DO
1394   0   42:3   169                           IF (LENGTH(DTID) <= 0) OR
1395   0   42:3   176                               ( NGTH(DTID) > TIDLENG) OR
```

```
1396   0   42:3   184                                (DLASTBLK < DFIRSTBLK) OR
1397   0   42:3   190                                (DLASTBYTE > FBLKSIZE) OR
1398   0   42:3   198                                (DLASTBYTE <= 0) OR
1399   0   42:3   204                                (DACCESS.YEAR >= 100) THEN
1400   0   42:4   215                              BEGIN OK := FALSE; DELENTRY(LINX,GDIRP) END
1401   0   42:3   223                            ELSE
1402   0   42:4   225                              LINX := LINX+1;
1403   0   42:1   232                        IF NOT OK THEN
1404   0   42:2   236                          BEGIN (*MUST HAVE BEEN CHANGED...WRITEIT*)
1405   0   42:3   236                            UNITWRITE(FUNIT,GDIRP^,
1406   0   42:3   240                                 (DNUMFILES+1)*SIZEOF(DIRENTRY),DIRBLK);
1407   0   42:3   251                            OK := IORSLT = INOERROR
1408   0   42:2   253                          END
1409   0   42:0   257                      END
1410   0   42:8   257                  END;
1411   0   42:6   257              IF OK THEN
1412   0   42:7   260                BEGIN UVID := DVID; UEOVBLK := DEOVBLK;
1413   0   42:8   272                  TIME(HNOW,DLOADTIME)
1414   0   42:7   279                END
1415   0   42:5   279            END;
1416   0   42:3   279        FETCHDIR := OK;
1417   0   42:3   282        IF NOT OK THEN
1418   0   42:4   286          BEGIN UVID := ''; UEOVBLK := MMAXINT;
1419   0   42:5   299            RELEASE(GDIRP); GDIRP := NIL
1420   0   42:4   307          END
1421   0   42:2   309      END
1422   0   42:0   309 END (*FETCHDIR*) ;
1423   0   42:0   328
1424   0   31:D     1 PROCEDURE WRITEDIR(*FUNIT: UNITNUM; FDIR: DIRP*);
1425   0   31:D     3   VAR HNOW,LNOW: INTEGER; OK: BOOLEAN; LDE: DIRENTRY;
1426   0   31:0     0 BEGIN
1427   0   31:1     0   WITH UNITABLE[FUNIT],FDIR^[0] DO
1428   0   31:2    14     BEGIN OK := (UVID = DVID) AND ((DFKIND = UNTYPEDFILE) OR
1429   0   31:3    31                                    (DFKIND = SECUREDIR));
1430   0   31:3    44         IF OK THEN
1431   0   31:4    47           BEGIN TIME(HNOW,LNOW);
1432   0   31:5    53             OK := (LNOW-DLOADTIME <= AGELIMIT) AND
1433   0   31:5    63                   ((LNOW-DLOADTIME) >= 0) AND
1434   0   31:5    72                      SYSCOM^.MISCINFO.HASCLOCK;
1435   0   31:5    83             IF NOT OK THEN
1436   0   31:6    87               BEGIN (*NO CLOCK OR TOO OLD*)
```

```
1437   0   31:7    87              UNITREAD(FUNIT,LDE,SIZEOF(DIRENTRY),DIRBLK);
1438   0   31:7    96              IF IORESULT = ORD(INOERROR) THEN
1439   0   31:8   102                OK := DVID = LDE.DVID;
1440   0   31:6   112            END;
1441   0   31:5   112          IF OK THEN
1442   0   31:6   115            BEGIN (*WE GUESS ALL IS SAFE...WRITEIT*)
1443   0   31:7   115              DFIRSTBLK := 0;   (*DIRTY FIX FOR YALOE BUGS*)
1444   0   31:7   119              UNITWRITE(FUNIT,FDIR^,
1445   0   31:7   122                        (DNUMFILES+1)*SIZEOF(DIRENTRY),DIRBLK);
1446   0   31:7   134              OK := IORESULT = ORD(INOERROR);
1447   0   31:7   140              IF DLASTBLK = 10 THEN (*REDUNDANT AFTERTHOUGHT*)
1448   0   31:8   147                UNITWRITE(FUNIT,FDIR^,
1449   0   31:8   150                          (DNUMFILES+1)*SIZEOF(DIRENTRY),6);
1450   0   31:7   162              IF OK THEN TIME(HNOW,DLOADTIME)
1451   0   31:6   173            END
1452   0   31:4   173        END;
1453   0   31:3   173      IF NOT OK THEN
1454   0   31:4   177        BEGIN UVID := ''; UEOVBLK := MMAXINT  END
1455   0   31:2   192    END
1456   0   31:0   192 END (*WRITEDIR*) ;
1457   0   31:0   204
1458   0   30:D     3 FUNCTION VOLSEARCH(*VAR FVID: VID; LOOKHARD: BOOLEAN; VAR FDIR: DIRP*);
1459   0   30:D     6   VAR LUNIT: UNITNUM; OK,PHYSUNIT: BOOLEAN; HNOW,LNOW: INTEGER;
1460   0   30:0     0 BEGIN VOLSEARCH := 0; FDIR := NIL;
1461   0   30:1     6   OK := FALSE; PHYSUNIT := FALSE;
1462   0   30:1    12   IF LENGTH(FVID) > 0 THEN
1463   0   30:2    19     BEGIN
1464   0   30:3    19       IF (FVID[1] = '#') AND (LENGTH(FVID) > 1) THEN
1465   0   30:4    32         BEGIN OK := TRUE;
1466   0   30:5    35           LUNIT := 0; HNOW := 2;
1467   0   30:5    41           REPEAT
1468   0   30:6    41             IF FVID[HNOW] IN DIGITS THEN
1469   0   30:7    53               LUNIT := LUNIT*10+ORD(FVID[HNOW])-ORD('0')
1470   0   30:6    61             ELSE OK := FALSE;
1471   0   30:6    69             HNOW := HNOW+1
1472   0   30:5    70           UNTIL (HNOW > LENGTH(FVID)) OR NOT OK;
1473   0   30:5    84           PHYSUNIT := OK AND (LUNIT > 0) AND (LUNIT <= MAXUNIT)
1474   0   30:4    92         END;
1475   0   30:3    95       IF NOT PHYSUNIT THEN
1476   0   30:4    99         BEGIN OK := FALSE; LUNIT := MAXUNIT;
1477   0   30:5   105           REPEAT
```

```
1478   0   30:6   105              OK := FVID = UNITABLE[LUNIT].UVID;
1479   0   30:6   116              IF NOT OK THEN LUNIT := LUNIT-1
1480   0   30:5   121            UNTIL OK OR (LUNIT = 0)
1481   0   30:4   129          END
1482   0   30:2   132      END;
1483   0   30:1   132    IF OK THEN
1484   0   30:2   135      IF UNITABLE[LUNIT].UISBLKD THEN
1485   0   30:3   144        WITH SYSCOM^ DO
1486   0   30:4   149          BEGIN OK := FALSE;  (*SEE IF GDIRP IS GOOD*)
1487   0   30:5   152            IF GDIRP <> NIL THEN
1488   0   30:6   158              IF FVID = GDIRP^[0].DVID THEN
1489   0   30:7   170                BEGIN TIME(HNOW,LNOW);
1490   0   30:8   176                  OK := LNOW-GDIRP^[0].DLOADTIME <= AGELIMIT
1491   0   30:7   185                END;
1492   0   30:5   191            IF NOT OK THEN
1493   0   30:6   195              BEGIN OK := PHYSUNIT;
1494   0   30:7   198                IF FETCHDIR(LUNIT) THEN
1495   0   30:8   205                  IF NOT PHYSUNIT THEN
1496   0   30:9   209                    OK := FVID = GDIRP^[0].DVID
1497   0   30:8   215                  ELSE
1498   0   30:7   223                ELSE
1499   0   30:8   225                  OK := IORESULT = ORD(INOERROR);[RELY ON IORESULT FROM FETCHDIR]
1500   0   30:6   231                END
1501   0   30:4   231          END;
1502   0   30:1   231    IF NOT OK AND LOOKHARD THEN
1503   0   30:2   237      BEGIN LUNIT := MAXUNIT; (*CHECK EACH DISK UNIT*)
1504   0   30:3   240        REPEAT
1505   0   30:4   240          WITH UNITABLE[LUNIT] DO
1506   0   30:5   248            IF UISBLKD THEN
1507   0   30:6   252              IF FETCHDIR(LUNIT) THEN
1508   0   30:7   259                OK := FVID = UVID;
1509   0   30:4   265          IF NOT OK THEN LUNIT := LUNIT-1
1510   0   30:3   270        UNTIL OK OR (LUNIT = 0)
1511   0   30:2   278      END;
1512   0   30:1   281    IF OK THEN
1513   0   30:2   284      WITH UNITABLE[LUNIT] DO
1514   0   30:3   292        BEGIN VOLSEARCH := LUNIT;
1515   0   30:4   295          IF LENGTH(UVID) > 0 THEN FVID := UVID;
1516   0   30:4   306          IF UISBLKD AND (SYSCOM^.GDIRP <> NIL) THEN
1517   0   30:5   317            BEGIN FDIR := SYSCOM^.GDIRP;
1518   0   30:6   323              TIME(HNOW,FDIR^[0].DLOADTIME)
```

```
1519    0    30:5    334            END
1520    0    30:3    334          END
1521    0    30:0    334 END (*VOLSEARCH*) ;
1522    0    30:0    352
1523    0    30:0    352
1524    0    30:0    352
1525    0    30:0    352 [$I SYSTEM.A ]
1525    0    30:0    352 [$I SYSTEM.B ]
1526    0    30:0    352
1527    0    30:0    352     (*****************************************************************)
1528    0    30:0    352     (*                                                               *)
1529    0    30:0    352     (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
1530    0    30:0    352     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
1531    0    30:0    352     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
1532    0    30:0    352     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
1533    0    30:0    352     (*                                                               *)
1534    0    30:0    352     (*****************************************************************)
1535    0    30:0    352
1536    0    30:0    352
1537    0    32:D      3 FUNCTION DIRSEARCH(*VAR FTID: TID; FINDPERM: BOOLEAN; FDIR: DIRP*);
1538    0    32:D      6   VAR I: DIRRANGE; FOUND: BOOLEAN;
1539    0    32:0      0 BEGIN DIRSEARCH := 0; FOUND := FALSE; I := 1;
1540    0    32:1      9   WHILE (I <= FDIR^[0].DNUMFILES) AND NOT FOUND DO
1541    0    32:2     22     BEGIN
1542    0    32:3     22       WITH FDIR^[I] DO
1543    0    32:4     28         IF DTID = FTID THEN
1544    0    32:5     36           IF FINDPERM = (DACCESS.YEAR <> 100) THEN
1545    0    32:6     49             BEGIN DIRSEARCH := I; FOUND := TRUE END;
1546    0    32:3     55       I := I+1
1547    0    32:2     56     END
1548    0    32:0     60 END (*DIRSEARCH*) ;
1549    0    32:0     76
1550    0    34:D      1 PROCEDURE DELENTRY(*FINX: DIRRANGE; FDIR: DIRP*);
1551    0    34:D      3   VAR I: DIRRANGE;
1552    0    34:0      0 BEGIN
1553    0    34:1      0   WITH FDIR^[0] DO
1554    0    34:2      6     BEGIN
1555    0    34:3      6       FOR I := FINX TO DNUMFILES-1 DO
1556    0    34:4     21         FDIR^[I] := FDIR^[I+1];
1557    0    34:3     40       FDIR^[DNUMFILES].DTID := '';
1558    0    34:3     53       DNUMFILES := DNUMFILES-1
```

```
1559   0   34:2    59      END
1560   0   34:0    62  END (*DELENTRY*) ;
1561   0   34:0    76
1562   0   35:0     1  PROCEDURE INSENTRY(*VAR FENTRY: DIRENTRY; FINX: DIRRANGE; FDIR: DIRP*);
1563   0   35:0     4    VAR I: DIRRANGE;
1564   0   35:0     0  BEGIN
1565   0   35:1     0    WITH FDIR^[0] DO
1566   0   35:2     6      BEGIN
1567   0   35:3     6        FOR I := DNUMFILES DOWNTO FINX DO
1568   0   35:4    19          FDIR^[I+1] := FDIR^[I];
1569   0   35:3    38        FDIR^[FINX] := FENTRY;
1570   0   35:3    45        DNUMFILES := DNUMFILES+1
1571   0   35:2    51      END
1572   0   35:0    54  END (*INSENTRY*) ;
1573   0   35:0    68
1574   0   47:0     3  FUNCTION ENTERTEMP(VAR FTID: TID; FSEGS: INTEGER;
1575   0   47:0     5                       FKIND: FILEKIND; FDIR: DIRP): DIRRANGE;
1576   0   47:0     7    VAR I,LASTI,DINX,SINX: DIRRANGE; RT11ISH: BOOLEAN;
1577   0   47:0    12        SSEGS: INTEGER; LDE: DIRENTRY;
1578   0   47:0    26
1579   0   48:0     1    PROCEDURE FINDMAX(CURINX: DIRRANGE; FIRSTOPEN,NEXTUSED: INTEGER);
1580   0   48:0     4      VAR FREEAREA: INTEGER;
1581   0   48:0     0    BEGIN
1582   0   48:1     0      FREEAREA := NEXTUSED-FIRSTOPEN;
1583   0   48:1     5      IF FREEAREA > FSEGS THEN
1584   0   48:2    10        BEGIN
1585   0   48:3    10          SINX := DINX; SSEGS := FSEGS;
1586   0   48:3    16          DINX := CURINX; FSEGS := FREEAREA
1587   0   48:2    19        END
1588   0   48:1    22      ELSE
1589   0   48:2    24        IF FREEAREA > SSEGS THEN
1590   0   48:3    29          BEGIN SSEGS := FREEAREA; SINX := CURINX END
1591   0   48:0    35    END (*FINDMAX*) ;
1592   0   48:0    48
1593   0   47:0     0  BEGIN (*ENTERTEMP*)
1594   0   47:1     0    DINX := 0; LASTI := FDIR^[0].DNUMFILES;
1595   0   47:1    11    SINX := 0; SSEGS := 0;
1596   0   47:1    17    IF FSEGS <= 0 THEN
1597   0   47:2    22      BEGIN RT11ISH := FSEGS < 0;
1598   0   47:3    27        FOR I := 1 TO LASTI DO
1599   0   47:4    39          FINDMAX(I,FDIR^[I-1].DLASTBLK,FDIR^[I].DFIRSTBLK);
```

```
1600   0   47:3    61       FINDMAX(LASTI+1,FDIR^[LASTI].DLASTBLK,FDIR^[0].DEOVBLK);
1601   0   47:3    76       IF RT11ISH THEN
1602   0   47:4    79         IF FSEGS DIV 2 <= SSEGS THEN
1603   0   47:5    86           BEGIN FSEGS := SSEGS; DINX := SINX END
1604   0   47:4    92         ELSE FSEGS := (FSEGS+1) DIV 2
1605   0   47:2    97     END
1606   0   47:1   101   ELSE
1607   0   47:2   103     BEGIN I := 1;
1608   0   47:3   106       WHILE I <= LASTI DO
1609   0   47:4   111         BEGIN
1610   0   47:5   111           IF FDIR^[I].DFIRSTBLK-FDIR^[I-1].DLASTBLK >= FSEGS THEN
1611   0   47:6   128             BEGIN DINX := I; I := LASTI END;
1612   0   47:5   134           I := I+1
1613   0   47:4   135         END;
1614   0   47:3   141       IF DINX = 0 THEN
1615   0   47:4   146         IF FDIR^[0].DEOVBLK-FDIR^[LASTI].DLASTBLK >= FSEGS THEN
1616   0   47:5   161           DINX := LASTI+1
1617   0   47:2   162     END;
1618   0   47:1   166   IF LASTI = MAXDIR THEN DINX := 0;
1619   0   47:1   174   IF DINX > 0 THEN
1620   0   47:2   179     BEGIN
1621   0   47:3   179       WITH LDE DO
1622   0   47:4   179         BEGIN
1623   0   47:5   179           DFIRSTBLK := FDIR^[DINX-1].DLASTBLK;
1624   0   47:5   188           DLASTBLK := DFIRSTBLK+FSEGS;
1625   0   47:5   193           DFKIND := FKIND; DTID := FTID;
1626   0   47:5   204           DLASTBYTE := FBLKSIZE;
1627   0   47:5   209           WITH DACCESS DO
1628   0   47:6   209             BEGIN MONTH := 0; DAY := 0; YEAR := 100 END
1629   0   47:4   227         END;
1630   0   47:3   227       INSENTRY(LDE,DINX,FDIR)
1631   0   47:2   231     END;
1632   0   47:1   233   ENTERTEMP := DINX
1633   0   47:0   233 END (*ENTERTEMP*) ;
1634   0   47:0   252
1635   0   47:0   252 (* FILE STATE HANDLERS *)
1636   0   47:0   252
1637   0    3:0     1 PROCEDURE FINIT(*VAR F: FIB; WINDOW: WINDOWP; RECWORDS: INTEGER*);
1638   0    3:0     0 BEGIN
1639   0    3:1     0   WITH F DO
1640   0    3:2     3     BEGIN FSTATE := FJANDW;
```

```
1641   0    3:3     8        FISOPEN := FALSE; FEOF := TRUE;
1642   0    3:3    18        FEOLN := TRUE; FWINDOW := WINDOW;
1643   0    3:3    26        IF (RECWORDS = 0) OR (RECWORDS = -2) THEN
1644   0    3:4    36          BEGIN
1645   0    3:5    36            FWINDOW^[1] := CHR(0); FRECSIZE := 1;
1646   0    3:5    46            IF RECWORDS = 0 THEN FSTATE := FNEEDCHAR
1647   0    3:4    54          END
1648   0    3:3    56        ELSE
1649   0    3:4    58          IF RECWORDS < 0 THEN
1650   0    3:5    63            BEGIN FWINDOW := NIL; FRECSIZE := 0 END
1651   0    3:4    71          ELSE FRECSIZE := RECWORDS+RECWORDS
1652   0    3:2    77      END
1653   0    3:0    80 END (*FINIT*) ;
1654   0    3:0    92
1655   0   49:D     1 PROCEDURE RESETER(VAR F:FIB);
1656   0   49:D     2   VAR BIGGER: BOOLEAN;
1657   0   49:0     0 BEGIN
1658   0   49:1     0   WITH F DO
1659   0   49:2     3       BEGIN FREPTCNT := 0;
1660   0   49:3     8          FEOLN := FALSE; FEOF := FALSE;
1661   0   49:3    18          IF FISBLKD THEN
1662   0   49:4    22            BEGIN BIGGER := FNXTBLK > FMAXBLK;
1663   0   49:5    31              IF BIGGER THEN FMAXBLK := FNXTBLK;
1664   0   49:5    41              IF FSOFTBUF THEN
1665   0   49:6    46                BEGIN
1666   0   49:7    46                  IF BIGGER THEN FMAXBYTE := FNXTBYTE
1667   0   49:7    52                  ELSE
1668   0   49:8    58                    IF FNXTBLK = FMAXBLK THEN
1669   0   49:9    67                      IF FNXTBYTE > FMAXBYTE THEN
1670   0   49:0    76                          BEGIN BIGGER := TRUE; FMAXBYTE := FNXTBYTE END;
1671   0   49:7    86                  IF FBUFCHNGD THEN
1672   0   49:8    91                    BEGIN FBUFCHNGD := FALSE; FMODIFIED := TRUE;
1673   0   49:9   101                      IF BIGGER THEN
1674   0   49:0   104                        FILLCHAR(FBUFFER[FNXTBYTE],FBLKSIZE-FNXTBYTE,0);
1675   0   49:9   120                      UNITWRITE(FUNIT,FBUFFER,FBLKSIZE,
1676   0   49:9   129                                FHEADER.DFIRSTBLK+FNXTBLK-1);
1677   0   49:9   141                      IF BIGGER AND (FHEADER.DFKIND = TEXTFILE)
1678   0   49:9   150                          AND ODD(FNXTBLK) THEN
1679   0   49:0   157                        BEGIN FMAXBLK := FMAXBLK+1;
1680   0   49:1   166                          FILLCHAR(FBUFFER,FBLKSIZE,0);
1681   0   49:1   176                          UNITWRITE(FUNIT,FBUFFER,FBLKSIZE,
```

```
1682   0    49:1   185                                                FHEADER.DFIRSTBLK+FNXTBLK)
1683   0    49:0   195                        END
1684   0    49:8   195                      END;
1685   0    49:7   195                    FNXTBYTE := FBLKSIZE
1686   0    49:6   198                  END;
1687   0    49:5   202                FNXTBLK := 0;
1688   0    49:5   207                IF FSOFTBUF AND (FHEADER.DFKIND = TEXTFILE) THEN
1689   0    49:6   221                    FNXTBLK := 2
1690   0    49:4   224              END
1691   0    49:2   226        END
1692   0    49:0   226 END (*RESETER*) ;
1693   0    49:0   242
1694   0     1:D     1 PROCEDURE FOPEN(*VAR F: FIB; VAR FTITLE: STRING;
1695   0     5:D     5                   FOPENOLD: BOOLEAN; JUNK PARAM*);
1696   0     5:D     5   LABEL 1;
1697   0     5:D     5   VAR LDIR: DIRP; LUNIT: UNITNUM; LINX: DIRRANGE;
1698   0     5:D     8       LSEGS,NBYTES: INTEGER; LKIND: FILEKIND;
1699   0     5:D    11       OLDHEAP: ^INTEGER; SWAPPED: BOOLEAN;
1700   0     5:D    13       SAVERSLT: IORSLTWD; LVID: VID; LTID: TID;
1701   0     5:0     0 BEGIN SYSCOM^.IORSLT := INOERROR;
1702   0     5:1     5   WITH F DO
1703   0     5:2     8     IF FISOPEN THEN SYSCOM^.IORSLT := INOTCLOSED
1704   0     5:2    16     ELSE
1705   0     5:3    20       IF SCANTITLE(FTITLE,LVID,LTID,LSEGS,LKIND) THEN
1706   0     5:4    35         BEGIN (*GOT AN OK TITLE*)
1707   0     5:5    35           IF ORD(FOPENOLD) > 1 THEN (*OLD CODE SPECIAL CASE*)
1708   0     5:6    40             FOPENOLD := (ORD(FOPENOLD) = 2) OR (ORD(FOPENOLD) = 4);
1709   0     5:5    49           SWAPPED := FALSE;
1710   0     5:5    52           WITH SWAPFIB^ DO
1711   0     5:6    57             IF FISOPEN AND (SYSCOM^.GDIRP = NIL) THEN
1712   0     5:7    69               BEGIN MARK(OLDHEAP);
1713   0     5:8    73                 NBYTES := ORD(SYSCOM^.LASTMP)-ORD(OLDHEAP);
1714   0     5:8    81                 IF (NBYTES > 0) AND (NBYTES < SIZEOF(DIRECTORY)+400) THEN
1715   0     5:9    96                   BEGIN
1716   0     5:0    96                     NBYTES := ORD(OLDHEAP)-ORD(EMPTYHEAP);
1717   0     5:0   103                     IF (NBYTES > 0) AND (NBYTES > SIZEOF(DIRECTORY)) AND
1718   0     5:0   112                         (UNITABLE[FUNIT].UVID = FVID) THEN
1719   0     5:1   129                       BEGIN
1720   0     5:2   129                         UNITWRITE(FUNIT,EMPTYHEAP^,SIZEOF(DIRECTORY),
1721   0     5:2   139                                   FHEADER.DFIRSTBLK);
1722   0     5:2   146                         RELEASE(EMPTYHEAP); SWAPPED := TRUE
```

```
1723   0    5:1   151                END
1724   0    5:9   154              END
1725   0    5:7   154          END;
1726   0    5:5   154        LUNIT := VOLSEARCH(LVID,TRUE,LDIR);
1727   0    5:5   165        IF LUNIT = 0 THEN SYSCOM^.IORSLT := INOUNIT
1728   0    5:5   173        ELSE
1729   0    5:6   177          WITH UNITABLE[LUNIT] DO
1730   0    5:7   185            BEGIN (*OK...OPEN UP FILE*)
1731   0    5:8   185              FISOPEN := TRUE; FMODIFIED := FALSE;
1732   0    5:8   197              FUNIT := LUNIT; FVID := LVID;
1733   0    5:8   211              FNXTBLK := 0; FISBLKD := UISBLKD;
1734   0    5:8   225              FSOFTBUF := UISBLKD AND (FRECSIZE <> 0);
1735   0    5:8   239              IF (LDIR <> NIL) AND (LENGTH(LTID) > 0) THEN
1736   0    5:9   251                BEGIN (*LOOKUP OR ENTER FHEADER IN DIRECTORY*)
1737   0    5:0   251                  LINX := DIRSEARCH(LTID,FOPENOLD,LDIR);
1738   0    5:0   261                  IF FOPENOLD THEN
1739   0    5:1   264                    IF LINX = 0 THEN
1740   0    5:2   269                      BEGIN SYSCOM^.IORSLT := INOFILE; GOTO 1 END
1741   0    5:1   276                    ELSE FHEADER := LDIR^[LINX]
1742   0    5:0   286                  ELSE (*OPEN NEW FILE*)
1743   0    5:1   290                    IF LINX > 0 THEN
1744   0    5:2   295                      BEGIN SYSCOM^.IORSLT := IDUPFILE; GOTO 1 END
1745   0    5:1   302                    ELSE
1746   0    5:2   304                      BEGIN (*MAKE A TEMP ENTRY*)
1747   0    5:3   304                        IF LKIND = UNTYPEDFILE THEN LKIND := DATAFILE;
1748   0    5:3   312                        LINX := ENTERTEMP(LTID,LSEGS,LKIND,LDIR);
1749   0    5:3   323                        IF (LINX > 0) AND (LKIND = TEXTFILE) THEN
1750   0    5:4   332                          WITH LDIR^[LINX] DO
1751   0    5:5   338                            BEGIN
1752   0    5:6   338                              IF ODD(DLASTBLK-DFIRSTBLK) THEN
1753   0    5:7   347                                DLASTBLK := DLASTBLK-1;
1754   0    5:6   357                              IF DLASTBLK-DFIRSTBLK < 4 THEN
1755   0    5:7   368                                BEGIN DELENTRY(LINX,LDIR); LINX := 0 END
1756   0    5:5   375                            END;
1757   0    5:3   375                        IF LINX = 0 THEN
1758   0    5:4   380                          BEGIN SYSCOM^.IORSLT := INOROOM; GOTO 1 END;
1759   0    5:3   387                        FHEADER := LDIR^[LINX]; FMODIFIED := TRUE;
1760   0    5:3   403                        WRITEDIR(LUNIT,LDIR)
1761   0    5:2   405                      END
1762   0    5:9   407                END
1763   0    5:8   407              ELSE (*FHEADER NOT IN DIRECTORY*)
```

```
1764   0    5:9   409                BEGIN
1765   0    5:0   409                  IF FOPENOLD AND (LENGTH(LTID) <> 0) THEN
1766   0    5:1   419                    BEGIN
1767   0    5:2   419                      SYSCOM^.IORSLT := INOFILE;
1768   0    5:2   424                      GOTO 1;
1769   0    5:1   426                      END;
1770   0    5:0   426                  WITH FHEADER DO
1771   0    5:1   432                    BEGIN (*DIRECT UNIT OPEN, SET UP DUMMY FHEADER*)
1772   0    5:2   432                      DFIRSTBLK := 0; DLASTBLK := MMAXINT;
1773   0    5:2   444                      IF UISBLKD THEN DLASTBLK := UEOVBLK;
1774   0    5:2   457                      DFKIND := LKIND; DTID := '';
1775   0    5:2   474                      DLASTBYTE := FBLKSIZE;
1776   0    5:2   482                      WITH DACCESS DO
1777   0    5:3   488                        BEGIN MONTH := 0; DAY := 0; YEAR := 0 END
1778   0    5:1   506                    END; [OF WITH]
1779   0    5:9   506                END;  [OF ELSE]
1780   0    5:8   506              IF FOPENOLD THEN
1781   0    5:9   509                FMAXBLK := FHEADER.DLASTBLK-FHEADER.DFIRSTBLK
1782   0    5:8   519              ELSE FMAXBLK := 0;
1783   0    5:8   531              IF FSOFTBUF THEN
1784   0    5:9   537                BEGIN
1785   0    5:0   537                  FNXTBYTE := FBLKSIZE; FBUFCHNGD := FALSE;
1786   0    5:0   551                  IF FOPENOLD THEN FMAXBYTE := FHEADER.DLASTBYTE
1787   0    5:0   560                  ELSE FMAXBYTE := FBLKSIZE;
1788   0    5:0   573                  WITH FHEADER DO
1789   0    5:1   579                    IF DFKIND = TEXTFILE THEN
1790   0    5:2   590                      BEGIN FNXTBLK := 2;
1791   0    5:3   596                        IF NOT FOPENOLD THEN
1792   0    5:4   600                          BEGIN (*NEW .TEXT, PUT NULLS IN FIRST PAGE*)
1793   0    5:5   600                            FILLCHAR(FBUFFER,SIZEOF(FBUFFER),0);
1794   0    5:5   611                            UNITWRITE(FUNIT,FBUFFER,FBLKSIZE,DFIRSTBLK);
1795   0    5:5   628                            UNITWRITE(FUNIT,FBUFFER,FBLKSIZE,DFIRSTBLK+1)
1796   0    5:4   647                          END
1797   0    5:2   647                      END
1798   0    5:9   647                END;
1799   0    5:6   647              IF FOPENOLD THEN FRESET(F)
1800   0    5:8   651              ELSE RESETER(F); (*NO GET!*)
1801   0    5:8   658    1:        IF SYSCOM^.IORSLT <> INOERROR THEN
1802   0    5:9   666                BEGIN FISOPEN := FALSE; FEOF := TRUE; FEOLN := TRUE END
1803   0    5:7   684            END;
1804   0    5:5   684          IF SWAPPED THEN
```

```
1805   0     5:6   687              BEGIN RELEASE(OLDHEAP); SYSCOM^.GDIRP := NIL;
1806   0     5:7   698                SAVERSLT := SYSCOM^.IORSLT;
1807   0     5:7   704                UNITREAD(SWAPFIB^.FUNIT,EMPTYHEAP^,SIZEOF(DIRECTORY),
1808   0     5:7   715                                  SWAPFIB^.FHEADER.DFIRSTBLK);
1809   0     5:7   723                SYSCOM^.IORSLT := SAVERSLT
1810   0     5:6   726              END
1811   0     5:4   728          END
1812   0     5:3   728        ELSE SYSCOM^.IORSLT := IBADTITLE
1813   0     5:0   733 END (*FOPEN*) ;
1814   0     5:0   758
1815   0     6:D     1 PROCEDURE FCLOSE(*VAR F: FIB; FTYPE: CLOSETYPE*);
1816   0     6:D     3   LABEL 1;
1817   0     6:D     3   VAR LINX,DUPINX: DIRRANGE; LDIR: DIRP; FOUND: BOOLEAN;
1818   0     6:0     0 BEGIN SYSCOM^.IORSLT := INOERROR;
1819   0     6:1     5   WITH F DO
1820   0     6:2     8     IF FISOPEN AND (FWINDOW <> SYSTERM^.FWINDOW) THEN
1821   0     6:3    20       BEGIN
1822   0     6:4    20         IF FISBLKD THEN
1823   0     6:5    24           WITH FHEADER DO
1824   0     6:6    29             IF LENGTH(DTID) > 0 THEN
1825   0     6:7    38               BEGIN (*FILE IN A DISK DIRECTORY...FIXUP MAYBE*)
1826   0     6:8    38                 IF FTYPE = CCRUNCH THEN
1827   0     6:9    43                   BEGIN FMAXBLK := FNXTBLK;
1828   0     6:0    50                     DACCESS.YEAR := 100; FTYPE := CLOCK;
1829   0     6:0    60                     IF FSOFTBUF THEN FMAXBYTE := FNXTBYTE
1830   0     6:9    68                   END;
1831   0     6:8    72                 RESETER(F);
1832   0     6:8    75                 IF FMODIFIED OR (DACCESS.YEAR = 100) OR (FTYPE = CPURGE) THEN
1833   0     6:9    93                   BEGIN (*HAVE TO CHANGE DIRECTORY ENTRY*)
1834   0     6:0    93                     IF FUNIT <> VOLSEARCH(FVID,FALSE,LDIR) THEN
1835   0     6:1   108                       BEGIN SYSCOM^.IORSLT := ILOSTUNIT; GOTO 1 END;
1836   0     6:0   115                     LINX := 1; FOUND := FALSE;
1837   0     6:0   121                     WHILE (LINX <= LDIR^[0].DNUMFILES) AND NOT FOUND DO
1838   0     6:1   134                       BEGIN (*LOOK FOR FIRST BLOCK MATCH*)
1839   0     6:2   134                         FOUND := (LDIR^[LINX].DFIRSTBLK = DFIRSTBLK) AND
1840   0     6:2   142                                  (LDIR^[LINX].DLASTBLK = DLASTBLK);
1841   0     6:2   153                         LINX := LINX + 1
1842   0     6:1   154                       END;
1843   0     6:0   160                     IF NOT FOUND THEN
1844   0     6:1   164                       BEGIN SYSCOM^.IORSLT := ILOSTFILE; GOTO 1 END;
1845   0     6:0   171                     LINX := LINX - 1; (*CORRECT OVERRUN*)
```

```
1846   0    6:0   176                    IF ((FTYPE = CNORMAL) AND (LDIR^[LINX].DACCESS.YEAR = 100))
1847   0    6:0   191                           OR (FTYPE = CPURGE) THEN
1848   0    6:1   197                       DELENTRY(LINX,LDIR)  (*ZAP FILE OUT OF EXISTANCE*)
1849   0    6:0   199                    ELSE
1850   0    6:1   203                      BEGIN (*WELL...LOCK IN A PERM DIR ENTRY*)
1851   0    6:2   203                        DUPINX := DIRSEARCH(DTID,TRUE,LDIR);
1852   0    6:2   214                        IF (DUPINX <> 0) AND (DUPINX <> LINX) THEN
1853   0    6:3   223                          BEGIN (*A DUPLICATE PERM ENTRY...ZAP OLD ONE*)
1854   0    6:4   223                            DELENTRY(DUPINX,LDIR);
1855   0    6:4   227                            IF DUPINX < LINX THEN LINX := LINX-1
1856   0    6:3   233                          END;
1857   0    6:2   237                        IF LDIR^[LINX].DACCESS.YEAR = 100 THEN
1858   0    6:3   250                          IF DACCESS.YEAR = 100 THEN
1859   0    6:4   260                            DACCESS := THEDATE
1860   0    6:3   263                          ELSE (*LEAVE ALONE...FILER SPECIAL CASE*)
1861   0    6:2   270                        ELSE
1862   0    6:3   272                          IF FMODIFIED AND (THEDATE.MONTH <> 0) THEN
1863   0    6:4   286                            DACCESS := THEDATE
1864   0    6:3   289                          ELSE
1865   0    6:4   296                            DACCESS := LDIR^[LINX].DACCESS;
1866   0    6:2   307                        DLASTBLK := DFIRSTBLK+FMAXBLK;
1867   0    6:2   317                        IF FSOFTBUF THEN DLASTBYTE := FMAXBYTE;
1868   0    6:2   329                        FHEADER.FILLER1 := 0;  [THIS HAD BETTER WORK, STEVE]
1869   0    6:2   336                        FMODIFIED := FALSE; LDIR^[LINX] := FHEADER
1870   0    6:1   345                      END;
1871   0    6:0   350                    WRITEDIR(FUNIT,LDIR)
1872   0    6:9   353                  END
1873   0    6:7   355                END;
1874   0    6:4   355              IF FTYPE = CPURGE THEN
1875   0    6:5   360                IF LENGTH(FHEADER.DTID) = 0 THEN
1876   0    6:6   369                  UNITABLE[FUNIT].UVID := '';
1877   0    6:4   381 1:           FEOF := TRUE; FEOLN := TRUE; FISOPEN := FALSE
1878   0    6:3   394         END;
1879   0    6:0   396 END (*FCLOSE*) ;
1880   0    6:0   422 [$I SYSTEM.B ]
1880   0    6:0   422 [$I SYSTEM.C ]
1881   0    6:0   422
1882   0    6:0   422      (*************************************************************)
1883   0    6:0   422      (*                                                           *)
1884   0    6:0   422      (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.   *)
1885   0    6:0   422      (*  PERMISSION TO COPY OR  'STRIBUTE THIS SOFTWARE OR DOCUMEN-    *)
```

```
1886   0     6:0   422      (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE   *)
1887   0     6:0   422      (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
1888   0     6:0   422      (*                                                               *)
1889   0     6:0   422      (*****************************************************************)
1890   0     6:0   422
1891   0     6:0   422  (* INPUT-OUTPUT PRIMITIVES *)
1892   0     6:0   422
1893   0     9:0     1  PROCEDURE XSEEK;
1894   0     9:0     0  BEGIN
1895   0     9:1     0    SYSCOM^.XEQERR := 11; [ NOT IMP ERR ]
1896   0     9:1     7    EXECERROR
1897   0     9:0     7  END (*XSEEK*) ;
1898   0     9:0    22
1899   0    14:0     1  PROCEDURE XREADREAL;
1900   0    14:0     0  BEGIN
1901   0    14:1     0    SYSCOM^.XEQERR := 11; [ NOT IMP ERR ]
1902   0    14:1     7    EXECERROR
1903   0    14:0     7  END (*XREADREAL*) ;
1904   0    14:0    22
1905   0    15:0     1  PROCEDURE XWRITEREAL;
1906   0    15:0     0  BEGIN
1907   0    15:1     0    SYSCOM^.XEQERR := 11; [ NOT IMP ERR ]
1908   0    15:1     7    EXECERROR
1909   0    15:0     7  END (*XWRITEREAL*) ;
1910   0    15:0    22
1911   0    50:0     3  FUNCTION CANTSTRETCH(VAR F: FIB): BOOLEAN; (*REPLACED BY RJH 2MAR78*)
1912   0    50:0     4    LABEL 1;
1913   0    50:0     4    VAR LINX: DIRRANGE; FOUND,OK: BOOLEAN; LAVAILBLK: INTEGER; LDIR: DIRP;
1914   0    50:0     0  BEGIN CANTSTRETCH := TRUE; OK := FALSE;
1915   0    50:0     6
1916   0    50:1     6    WITH F,FHEADER DO
1917   0    50:2    14      IF LENGTH(DTID) > 0 THEN
1918   0    50:3    23        BEGIN (*IN A DIRECTORY FOR SURE*)
1919   0    50:4    23          IF FUNIT <> VOLSEARCH(FVID,FALSE,LDIR) THEN
1920   0    50:5    38            BEGIN SYSCOM^.IORSLT := ILOSTUNIT; GOTO 1 END;
1921   0    50:4    45          FOUND := FALSE; LINX := 1;
1922   0    50:4    51          WHILE (LINX <= LDIR^[0].DNUMFILES) AND NOT FOUND DO
1923   0    50:5    64            BEGIN
1924   0    50:6    64              FOUND := (LDIR^[LINX].DFIRSTBLK = DFIRSTBLK) AND
1925   0    50:6    72                       (LDIR^[LINX].DLASTBLK = DLASTBLK);
1926   0    50:6    83              LINX := LINX+1
```

```
1927   0   50:5    84            END;
1928   0   50:4    90          IF NOT FOUND THEN
1929   0   50:5    94            BEGIN SYSCOM^.IORSLT := ILOSTFILE; GOTO 1 END;
1930   0   50:4   101          IF LINX > LDIR^[0].DNUMFILES THEN
1931   0   50:5   111            LAVAILBLK := LDIR^[0].DEOVBLK
1932   0   50:4   115          ELSE LAVAILBLK := LDIR^[LINX].DFIRSTBLK;
1933   0   50:4   127          IF (DLASTBLK < LAVAILBLK) OR (DLASTBYTE < FBLKSIZE) THEN
1934   0   50:5   141            BEGIN
1935   0   50:6   141              WITH LDIR^[LINX-1] DO
1936   0   50:7   149                BEGIN
1937   0   50:8   149                  DLASTBLK := LAVAILBLK; DLASTBYTE := FBLKSIZE;
1938   0   50:8   161                  WRITEDIR(FUNIT,LDIR);
1939   0   50:8   166                  IF IORESULT <> ORD(INOERROR) THEN GOTO 1
1940   0   50:7   174                END;
1941   0   50:6   174              FEOF := FALSE; FEOLN := FALSE;
1942   0   50:6   184              IF FSTATE <> FJANDW THEN FSTATE := FNEEDCHAR; (*RJH 2MAR78*)
1943   0   50:6   195              DLASTBLK := LAVAILBLK; DLASTBYTE := FBLKSIZE;
1944   0   50:6   207              DACCESS.YEAR := 100; CANTSTRETCH := FALSE
1945   0   50:5   214            END;
1946   0   50:4   217          OK := TRUE;
1947   0   50:3   220        END;
1948   0   50:1   220 1:  IF NOT OK THEN
1949   0   50:2   224        BEGIN F.FEOF := TRUE; F.FEOLN := TRUE END
1950   0   50:0   234 END (*CANTSTRETCH*) ;
1951   0   50:0   252
1952   0    4:D     1 PROCEDURE FRESET(*VAR F: FIB*);
1953   0    4:0     0 BEGIN SYSCOM^.IORSLT := INOERROR;
1954   0    4:1     5   WITH F DO
1955   0    4:2     8     IF FISOPEN THEN
1956   0    4:3    12       BEGIN RESETER(F);
1957   0    4:4    15         IF FRECSIZE > 0 THEN
1958   0    4:5    21           IF FSTATE = FJANDW THEN FGET(F)
1959   0    4:5    28           ELSE FSTATE := FNEEDCHAR
1960   0    4:3    35       END
1961   0    4:0    37 END (*FRESET*) ;
1962   0    4:0    50
1963   0    1:D     3 FUNCTION FBLOCKIO(*VAR F: FIB; VAR A: WINDOW; I: INTEGER;
1964   0   28:D     9                     NBLOCKS,RBLOCK: INTEGER; DOREAD: BOOLEAN*);
1965   0   28:0     0 BEGIN FBLOCKIO := 0; SYSCOM^.IORSLT := INOERROR;
1966   0   28:1     8   WITH F DO
1967   0   28:2    11     IF FISOPEN AND (NBLOCKS >
```

```
1968   0   28:3    19        IF FISBLKD THEN
1969   0   28:4    23          WITH FHEADER DO
1970   0   28:5    28            BEGIN
1971   0   28:6    28              IF RBLOCK < 0 THEN RBLOCK := FNXTBLK;
1972   0   28:6    38              RBLOCK := DFIRSTBLK+RBLOCK;
1973   0   28:6    44              IF RBLOCK+NBLOCKS > DLASTBLK THEN
1974   0   28:7    52                IF NOT DOREAD THEN
1975   0   28:8    56                  IF CANTSTRETCH( F ) THEN;
1976   0   28:6    63              IF RBLOCK+NBLOCKS > DLASTBLK THEN
1977   0   28:7    71                NBLOCKS := DLASTBLK-RBLOCK;
1978   0   28:6    77              FEOF := RBLOCK >= DLASTBLK;
1979   0   28:6    85              IF NOT FEOF THEN
1980   0   28:7    90                BEGIN
1981   0   28:8    90                  IF DOREAD THEN
1982   0   28:9    93                    UNITREAD(FUNIT,A[1],NBLOCKS*FBLKSIZE,RBLOCK)
1983   0   28:8   106                   ELSE
1984   0   28:9   108                     BEGIN FMODIFIED := TRUE;
1985   0   28:0   113                       UNITWRITE(FUNIT,A[1],NBLOCKS*FBLKSIZE,RBLOCK)
1986   0   28:9   126                     END;
1987   0   28:8   126                   FBLOCKIO := NBLOCKS;
1988   0   28:8   129                   RBLOCK := RBLOCK+NBLOCKS;
1989   0   28:8   134                   FEOF := RBLOCK = DLASTBLK;
1990   0   28:8   142                   FNXTBLK := RBLOCK-DFIRSTBLK;
1991   0   28:8   150                   IF FNXTBLK > FMAXBLK THEN FMAXBLK := FNXTBLK
1992   0   28:7   162                 END
1993   0   28:5   166            END
1994   0   28:3   166      ELSE
1995   0   28:4   168        BEGIN FBLOCKIO := NBLOCKS;
1996   0   28:5   171          IF DOREAD THEN
1997   0   28:6   174            UNITREAD(FUNIT,A[1],NBLOCKS*FBLKSIZE,RBLOCK)
1998   0   28:5   187          ELSE
1999   0   28:6   189            UNITWRITE(FUNIT,A[1],NBLOCKS*FBLKSIZE,RBLOCK);
2000   0   28:5   202          IF IORESULT = ORD(INOERROR) THEN
2001   0   28:6   208            IF DOREAD THEN
2002   0   28:7   211              BEGIN RBLOCK := NBLOCKS*FBLKSIZE;
2003   0   28:8   218                RBLOCK := RBLOCK+SCAN(-RBLOCK,<>CHR(0),A[1+RBLOCK-1]);
2004   0   28:8   235                RBLOCK := (RBLOCK+FBLKSIZE-1) DIV FBLKSIZE;
2005   0   28:8   248                FBLOCKIO := RBLOCK;
2006   0   28:8   251                FEOF := RBLOCK < NBLOCKS
2007   0   28:7   255              END
2008   0   28:6   258            ELSE
```

```
2009   0   28:5   260            ELSE FBLOCKIO := 0
2010   0   28:4   262          END
2011   0   28:2   265      ELSE
2012   0   28:3   267        SYSCOM^.IORSLT := INOTOPEN
2013   0   28:0   270 END (*FBLOCKIO*) ;
2014   0   28:0   288
2015   0    7:D     1 PROCEDURE FGET(*VAR F: FIB*);
2016   0    7:D     2   LABEL 1, 2;
2017   0    7:D     2   VAR LEFTOGET,WININX,LEFTINBUF,AMOUNT: INTEGER;
2018   0    7:D     6       DONE: BOOLEAN;
2019   0    7:0     0 BEGIN SYSCOM^.IORSLT := INOERROR;
2020   0    7:1     5   WITH F DO
2021   0    7:2     8     IF FISOPEN THEN
2022   0    7:3    12       BEGIN
2023   0    7:4    12         IF FREPTCNT > 0 THEN
2024   0    7:5    19           BEGIN FREPTCNT := FREPTCNT-1; IF FREPTCNT > 0 THEN GOTO 2 END;
2025   0    7:4    37         IF FSOFTBUF THEN
2026   0    7:5    42           WITH FHEADER DO
2027   0    7:6    47             BEGIN
2028   0    7:7    47               LEFTOGET := FRECSIZE; WININX := 0;
2029   0    7:7    54               REPEAT
2030   0    7:8    54                 IF FNXTBLK = FMAXBLK THEN
2031   0    7:9    63                   IF FNXTBYTE+LEFTOGET > FMAXBYTE THEN GOTO 1
2032   0    7:9    76                   ELSE LEFTINBUF := DLASTBYTE-FNXTBYTE
2033   0    7:8    81                 ELSE LEFTINBUF := FBLKSIZE-FNXTBYTE;
2034   0    7:8    98                 AMOUNT := LEFTOGET;
2035   0    7:8   101                 IF AMOUNT > LEFTINBUF THEN AMOUNT := LEFTINBUF;
2036   0    7:8   109                 IF AMOUNT > 0 THEN
2037   0    7:9   114                   BEGIN
2038   0    7:0   114                     MOVELEFT(FBUFFER[FNXTBYTE],FWINDOW^[WININX],AMOUNT);
2039   0    7:0   126                     FNXTBYTE := FNXTBYTE+AMOUNT;
2040   0    7:0   135                     WININX := WININX+AMOUNT;
2041   0    7:0   140                     LEFTOGET := LEFTOGET-AMOUNT
2042   0    7:9   141                   END;
2043   0    7:8   145                 DONE := LEFTOGET = 0;
2044   0    7:8   150                 IF NOT DONE THEN
2045   0    7:9   154                   BEGIN
2046   0    7:0   154                     IF FBUFCHNGD THEN
2047   0    7:1   159                       BEGIN FBUFCHNGD := FALSE; FMODIFIED := TRUE;
2048   0    7:2   169                         UNITWRITE(FUNIT,FBUFFER,FBLKSIZE,DFIRSTBLK+FNXTBLK-1)
2049   0    7:1   189                       END;
```

```
2050   0    7:0   189                    IF IORESULT <> ORD(INOERROR) THEN GOTO 1;
2051   0    7:0   197                    UNITREAD(FUNIT,FBUFFER,FBLKSIZE,DFIRSTBLK+FNXTBLK);
2052   0    7:0   215                    IF IORESULT <> ORD(INOERROR) THEN GOTO 1;
2053   0    7:0   223                    FNXTBLK := FNXTBLK+1; FNXTBYTE := 0
2054   0    7:9   235                  END
2055   0    7:7   237                UNTIL DONE
2056   0    7:6   237              END
2057   0    7:4   240          ELSE
2058   0    7:5   242            BEGIN
2059   0    7:6   242              UNITREAD(FUNIT,FWINDOW^,FRECSIZE);
2060   0    7:6   253              IF IORESULT <> ORD(INOERROR) THEN GOTO 1
2061   0    7:5   261            END;
2062   0    7:4   261          IF FRECSIZE = 1 THEN (*FILE OF CHAR*)
2063   0    7:5   267            BEGIN FEOLN := FALSE;
2064   0    7:6   272              IF FSTATE <> FJANDW THEN FSTATE := FGOTCHAR;
2065   0    7:6   283              IF FWINDOW^[0] = CHR(EOL) THEN
2066   0    7:7   291                BEGIN FWINDOW^[0] := ' '; FEOLN := TRUE; GOTO 2 END;
2067   0    7:6   303              IF FWINDOW^[0] = CHR(DLE) THEN
2068   0    7:7   311                BEGIN FGET(F);
2069   0    7:8   314                  AMOUNT := ORD(FWINDOW^[0])-32;
2070   0    7:8   322                  IF (AMOUNT > 0) AND (AMOUNT <= 127) THEN
2071   0    7:9   331                    BEGIN
2072   0    7:0   331                      FWINDOW^[0] := ' ';
2073   0    7:0   336                      FREPTCNT := AMOUNT;
2074   0    7:0   341                      GOTO 2
2075   0    7:9   343                    END;
2076   0    7:8   343                  FGET(F)
2077   0    7:7   344                END;
2078   0    7:6   346              IF FWINDOW^[0] = CHR(0) THEN
2079   0    7:7   354                BEGIN (*EOF HANDLING*)
2080   0    7:8   354                  IF FSOFTBUF AND (FHEADER.DFKIND = TEXTFILE) THEN
2081   0    7:9   368                    BEGIN (*END OF 2 BLOCK PAGE*)
2082   0    7:0   368                      IF ODD(FNXTBLK) THEN FNXTBLK := FNXTBLK+1;
2083   0    7:0   382                      FNXTBYTE := FBLKSIZE; FGET(F)
2084   0    7:9   390                    END
2085   0    7:8   392                  ELSE
2086   0    7:9   394                    BEGIN FWINDOW^[0] := ' '; GOTO 1 END
2087   0    7:7   401                END
2088   0    7:5   401            END
2089   0    7:3   401        END
2090   0    7:2   401      ELSE
```

```
2091   0    7:3   403        BEGIN
2092   0    7:4   403          SYSCOM^.IORSLT := INOTOPEN;
2093   0    7:4   408  1:      FEOF := TRUE; FEOLN := TRUE
2094   0    7:3   416        END;
2095   0    7:1   418  2:
2096   0    7:0   418  END (*FGET*) ;
2097   0    7:0   442
2098   0    8:0     1  PROCEDURE FPUT(*VAR F: FIB*);
2099   0    8:0     2    LABEL 1;
2100   0    8:0     2    VAR LEFTOPUT,WININX,LEFTINBUF,AMOUNT: INTEGER;
2101   0    8:0     6        DONE: BOOLEAN;
2102   0    8:0     0  BEGIN SYSCOM^.IORSLT := INOERROR;
2103   0    8:1     5    WITH F DO
2104   0    8:2     8      IF FISOPEN THEN
2105   0    8:3    12        BEGIN
2106   0    8:4    12          IF FSOFTBUF THEN
2107   0    8:5    17            WITH FHEADER DO
2108   0    8:6    22              BEGIN
2109   0    8:7    22                LEFTOPUT := FRECSIZE; WININX := 0;
2110   0    8:7    29                REPEAT
2111   0    8:8    29                  IF DFIRSTBLK+FNXTBLK = DLASTBLK THEN
2112   0    8:9    40                    IF FNXTBYTE+LEFTOPUT > DLASTBYTE THEN
2113   0    8:0    51                      IF CANTSTRETCH( F ) THEN
2114   0    8:1    58                          BEGIN SYSCOM^.IORSLT := INOROOM; GOTO 1 END
2115   0    8:0    65                      ELSE LEFTINBUF := FBLKSIZE-FNXTBYTE
2116   0    8:9    70                    ELSE LEFTINBUF := DLASTBYTE-FNXTBYTE
2117   0    8:8    81                  ELSE LEFTINBUF := FBLKSIZE-FNXTBYTE;
2118   0    8:8    98                  AMOUNT := LEFTOPUT;
2119   0    8:8   101                  IF AMOUNT > LEFTINBUF THEN AMOUNT := LEFTINBUF;
2120   0    8:8   109                  IF AMOUNT > 0 THEN
2121   0    8:9   114                    BEGIN FBUFCHNGD := TRUE;
2122   0    8:0   119                      MOVELEFT(FWINDOW^[WININX],FBUFFER[FNXTBYTE],AMOUNT);
2123   0    8:0   131                      FNXTBYTE := FNXTBYTE+AMOUNT;
2124   0    8:0   140                      WININX := WININX+AMOUNT;
2125   0    8:0   145                      LEFTOPUT := LEFTOPUT-AMOUNT
2126   0    8:9   146                    END;
2127   0    8:8   150                  DONE := LEFTOPUT = 0;
2128   0    8:8   155                  IF NOT DONE THEN
2129   0    8:9   159                    BEGIN
2130   0    8:0   159                      IF FBUFCHNGD THEN
2131   0    8:1   164                        BEGIN FB  CHNGD := FALSE; FMODIFIED := TRUE;
```

```
2132  0    8:2   174                          UNITWRITE(FUNIT,FBUFFER,FBLKSIZE,DFIRSTBLK+FNXTBLK-1)
2133  0    8:1   194                        END;
2134  0    8:0   194                      IF IORESULT <> ORD(INOERROR) THEN GOTO 1;
2135  0    8:0   202                      IF FNXTBLK < FMAXBLK THEN
2136  0    8:1   211                        UNITREAD(FUNIT,FBUFFER,FBLKSIZE,DFIRSTBLK+FNXTBLK)
2137  0    8:0   229                      ELSE
2138  0    8:1   231                        FILLCHAR(FBUFFER,FBLKSIZE,CHR(0));
2139  0    8:0   241                      IF IORESULT <> ORD(INOERROR) THEN GOTO 1;
2140  0    8:0   249                      FNXTBLK := FNXTBLK+1; FNXTBYTE := 0
2141  0    8:9   261                    END
2142  0    8:7   263                UNTIL DONE;
2143  0    8:7   266                IF FRECSIZE = 1 THEN
2144  0    8:8   272                  IF FWINDOW^[0] = CHR(EOL) THEN
2145  0    8:9   280                    IF DFKIND = TEXTFILE THEN
2146  0    8:0   290                      IF (FNXTBYTE >= FBLKSIZE-127) AND NOT ODD(FNXTBLK) THEN
2147  0    8:1   306                        BEGIN
2148  0    8:2   306                          FNXTBYTE := FBLKSIZE-1;
2149  0    8:2   315                          FWINDOW^[0] := CHR(0);
2150  0    8:2   320                          FPUT(F)
2151  0    8:1   321                        END
2152  0    8:6   323                END
2153  0    8:4   323              ELSE
2154  0    8:5   325                BEGIN
2155  0    8:6   325                  UNITWRITE(FUNIT,FWINDOW^,FRECSIZE);
2156  0    8:6   336                  IF IORESULT <> ORD(INOERROR) THEN GOTO 1
2157  0    8:5   344                END
2158  0    8:3   344            END
2159  0    8:2   344          ELSE
2160  0    8:3   346            BEGIN
2161  0    8:4   346              SYSCOM^.IORSLT := INOTOPEN;
2162  0    8:4   351 1:           FEOF := TRUE; FEOLN := TRUE
2163  0    8:3   359            END
2164  0    8:0   361 END (*FPUT*) ;
2165  0    8:0   382
2166  0   10:D     3 FUNCTION FEOF(*VAR F: FIB*);
2167  0   10:0     0 BEGIN FEOF := F.FEOF END;
2168  0   10:0    16
2169  0   10:0    16 (* TEXT FILE INTRINSICS *)
2170  0   10:0    16
2171  0   11:D     3 FUNCTION FEOLN(*VAR F: FIB*);
2172  0   11:0     0 BEGIN FEOLN := F.FEOLN END;
```

```
2173   0   11:0    16
2174   0   22:0     1 PROCEDURE FWRITELN(*VAR F: FIB*);
2175   0   22:0     0 BEGIN
2176   0   22:1     0   F.FWINDOW^[0] := CHR(EOL); FPUT(F)
2177   0   22:0     6 END (*FWRITELN*) ;
2178   0   22:0    20
2179   0   17:0     1 PROCEDURE FWRITECHAR(*VAR F: FIB; CH: CHAR; RLENG: INTEGER*);
2180   0   17:0     4   LABEL  1;
2181   0   17:0     0 BEGIN
2182   0   17:1     0   WITH F DO
2183   0   17:2     3     IF FISOPEN THEN
2184   0   17:3     7       IF FSOFTBUF THEN
2185   0   17:4    12         BEGIN
2186   0   17:5    12           WHILE RLENG > 1 DO
2187   0   17:6    17             BEGIN FWINDOW^[0] := ' '; FPUT(F);
2188   0   17:7    25               RLENG := RLENG-1
2189   0   17:6    26             END;
2190   0   17:5    32           FWINDOW^[0] := CH; FPUT(F)
2191   0   17:4    38         END
2192   0   17:3    40       ELSE
2193   0   17:4    42         BEGIN
2194   0   17:5    42           WHILE RLENG > 1 DO
2195   0   17:6    47             BEGIN FWINDOW^[0] := ' ';
2196   0   17:7    52               UNITWRITE(FUNIT,FWINDOW^,1);
2197   0   17:7    62               RLENG := RLENG-1
2198   0   17:6    63             END;
2199   0   17:5    69           FWINDOW^[0] := CH;
2200   0   17:5    74           UNITWRITE(FUNIT,FWINDOW^,1)
2201   0   17:4    84         END
2202   0   17:2    84     ELSE SYSCOM^.IORSLT := INOTOPEN;
2203   0   17:1    91 1:
2204   0   17:0    91 END (*FWRITECHAR*) ;
2205   0   17:0   108
2206   0   13:0     1 PROCEDURE FWRITEINT(*VAR F: FIB; I,RLENG: INTEGER*);
2207   0   13:0     4   LABEL 1;
2208   0   13:0     4   VAR POT,COL: INTEGER; CH: CHAR;
2209   0   13:0     7       SUPPRESSING: BOOLEAN; S: STRING[10];
2210   0   13:0     0 BEGIN COL := 1;
2211   0   13:1     3   S[0] := CHR(10); SUPPRESSING := TRUE;
2212   0   13:1    11   IF I < 0 THEN
2213   0   13:2    16     BEGIN I := ABS(I); S[1] :=  -'; COL := 2;
```

```
2214   0    13:3    26        IF I = 0 THEN (*HARDWARE SPECIAL CASE*)
2215   0    13:4    33          BEGIN S := '-32768'; GOTO 1 END
2216   0    13:2    48      END;
2217   0    13:1    48    FOR POT := 4 DOWNTO 0 DO
2218   0    13:2    59      BEGIN CH := CHR(I DIV IPOT[POT] + ORD('0'));
2219   0    13:3    72        IF (CH = '0') AND (POT > 0) AND SUPPRESSING THEN
2220   0    13:3    83        ELSE (*FORMAT THE CHAR*)
2221   0    13:4    35          BEGIN SUPPRESSING := FALSE;
2222   0    13:5    88            S[COL] := CH; COL := COL+1;
2223   0    13:5    98            IF CH <> '0' THEN I := I MOD IPOT[POT]
2224   0    13:4   110          END
2225   0    13:2   114      END;
2226   0    13:1   121    S[0] := CHR(COL-1);
2227   0    13:1   128 1:IF RLENG < LENGTH(S) THEN
2228   0    13:2   136      RLENG := LENGTH(S);
2229   0    13:1   142    FWRITESTRING(F,S,RLENG)
2230   0    13:0   146 END (*FWRITEINT*) ;
2231   0    13:0   162
2232   0    19:D     1 PROCEDURE FWRITESTRING(*VAR F: FIB; VAR S: STRING; RLENG: INTEGER*);
2233   0    19:D     4   VAR SINX: INTEGER;
2234   0    19:0     0 BEGIN
2235   0    19:1     0   WITH F DO
2236   0    19:2     3     IF FISOPEN THEN
2237   0    19:3     7       BEGIN
2238   0    19:4     7         IF RLENG <= 0 THEN RLENG := LENGTH(S);
2239   0    19:4    17         IF RLENG > LENGTH(S) THEN
2240   0    19:5    24           BEGIN FWRITECHAR(F,' ',RLENG-LENGTH(S)); RLENG := LENGTH(S) END;
2241   0    19:4    38         IF FSOFTBUF THEN
2242   0    19:5    43           BEGIN SINX := 1;
2243   0    19:6    46             WHILE (SINX <= RLENG) AND NOT FEOF DO
2244   0    19:7    55               BEGIN FWINDOW^[0] := S[SINX]; FPUT(F); SINX := SINX+1 END
2245   0    19:5    70           END
2246   0    19:4    72         ELSE
2247   0    19:5    74           UNITWRITE(FUNIT,S[1],RLENG)
2248   0    19:3    83       END
2249   0    19:2    83     ELSE SYSCOM^.IORSLT := INOTOPEN
2250   0    19:0    88 END (*FWRITESTRING*) ;
2251   0    19:0   104
2252   0    18:D     1 PROCEDURE FREADSTRING(*VAR F: FIB; VAR S: STRING; SLENG: INTEGER*);
2253   0    18:D     4   VAR SINX: INTEGER; CH: CHAR;
2254   0    18:0     0 BEGIN
```

```
2255   0   16:1     0    WITH F DO
2256   0   16:2     3        BEGIN SINX := 1;
2257   0   16:3     6          IF FSTATE = FNEEDCHAR THEN FGET(F);
2258   0   16:3    15          S[0] := CHR(SLENG); (*NO INV INDEX*)
2259   0   16:3    19          WHILE (SINX <= SLENG) AND NOT (FEOLN OR FEOF) DO
2260   0   16:4    31            BEGIN CH := FWINDOW^[0];
2261   0   16:5    37              IF FUNIT = 1 THEN
2262   0   16:6    43                IF CHECKDEL(CH,SINX) THEN
2263   0   16:6    52                ELSE
2264   0   16:7    54                  BEGIN S[SINX] := CH; SINX := SINX + 1 END
2265   0   16:5    63              ELSE
2266   0   16:6    65                BEGIN S[SINX] := CH; SINX := SINX + 1 END;
2267   0   16:5    74              FGET(F)
2268   0   16:4    75            END;
2269   0   16:3    79          S[0] := CHR(SINX - 1);
2270   0   16:3    85          WHILE NOT FEOLN DO FGET(F)
2271   0   16:2    91        END
2272   0   16:0    95 END (*FREADSTRING*) ;
2273   0   16:0   112
2274   0   20:0     1 PROCEDURE FWRITEBYTES(*VAR F: FIB; VAR A: WINDOW; RLENG,ALENG: INTEGER*);
2275   0   20:0     5   VAR AINX: INTEGER;
2276   0   20:0     0 BEGIN
2277   0   20:1     0   WITH F DO
2278   0   20:2     3     IF FISOPEN THEN
2279   0   20:3     7       BEGIN
2280   0   20:4     7         IF RLENG > ALENG THEN
2281   0   20:5    12           BEGIN FWRITECHAR(F,' ',RLENG-ALENG); RLENG := ALENG END;
2282   0   20:4    22         IF FSOFTBUF THEN
2283   0   20:5    27           BEGIN AINX := 0;
2284   0   20:6    30             WHILE (AINX < RLENG) AND NOT FEOF DO
2285   0   20:7    39               BEGIN FWINDOW^[0] := A[AINX]; FPUT(F); AINX := AINX+1 END
2286   0   20:5    54           END
2287   0   20:4    56         ELSE
2288   0   20:5    58           UNITWRITE(FUNIT,A,RLENG)
2289   0   20:3    67       END
2290   0   20:2    67     ELSE SYSCOM^.IORSLT := INOTOPEN
2291   0   20:0    72 END (*FWRITEBYTES*) ;
2292   0   20:0    88
2293   0   21:0     1 PROCEDURE FREADLN(*VAR F: FIB*);
2294   0   21:0     0 BEGIN
2295   0   21:1     0   WHILE NOT F.FEOLN DO FGET(F)
```

```
2296   0   21:1    10   IF F.FSTATE = FJANDW THEN FGET(F)
2297   0   21:1    17   ELSE
2298   0   21:2    21     BEGIN F.FSTATE := FNEEDCHAR; F.FEOLN := FALSE END
2299   0   21:0    31 END (*FREADLN*) ;
2300   0   21:0    46
2301   0   16:D     1 PROCEDURE FREADCHAR(*VAR F: FIB; VAR CH: CHAR*);
2302   0   16:0     0 BEGIN
2303   0   16:1     0   WITH F DO
2304   0   16:2     3       BEGIN SYSCOM^.IORSLT := INOERROR;
2305   0   16:3     8         IF FSTATE = FNEEDCHAR THEN FGET(F);
2306   0   16:3    17         CH := FWINDOW^[0];
2307   0   16:3    23         IF FSTATE = FJANDW THEN FGET(F)
2308   0   16:3    30         ELSE FSTATE := FNEEDCHAR
2309   0   16:2    37       END
2310   0   16:0    39 END (*FREADCHAR*) ;
2311   0   16:0    52
2312   0   12:D     1 PROCEDURE FREADINT(*VAR F: FIB; VAR I: INTEGER*);
2313   0   12:D     3   LABEL 1;
2314   0   12:D     3   VAR CH: CHAR; NEG,IVALID: BOOLEAN; SINX: INTEGER;
2315   0   12:0     0 BEGIN
2316   0   12:1     0   WITH F DO
2317   0   12:2     3       BEGIN I := 0; NEG := FALSE; IVALID := FALSE;
2318   0   12:3    12         IF FSTATE = FNEEDCHAR THEN FGET(F);
2319   0   12:3    21         WHILE (FWINDOW^[0] = ' ') AND NOT FEOF DO FGET(F);
2320   0   12:3    38         IF FEOF THEN GOTO 1;
2321   0   12:3    44         CH := FWINDOW^[0];
2322   0   12:3    50         IF (CH = '+') OR (CH = '-') THEN
2323   0   12:4    59           BEGIN NEG := CH = '-'; FGET(F); CH := FWINDOW^[0] END;
2324   0   12:3    73         IF CH IN DIGITS THEN
2325   0   12:4    83           BEGIN  IVALID := TRUE; SINX := 1;
2326   0   12:5    89             REPEAT
2327   0   12:6    89               I := I*10+ORD(CH)-ORD('0');
2328   0   12:6    99               FGET(F); CH := FWINDOW^[0]; SINX := SINX+1;
2329   0   12:6   113               IF FUNIT = 1 THEN
2330   0   12:7   119                 WHILE CHECKDEL(CH,SINX) DO
2331   0   12:8   128                   BEGIN
2332   0   12:9   128                     IF SINX = 1 THEN I := 0 ELSE I := I DIV 10;
2333   0   12:9   144                     FGET(F); CH := FWINDOW^[0]
2334   0   12:8   150                   END
2335   0   12:5   153             UNTIL NOT (CH IN DIGITS) OR FEOLN
2336   0   12:4   164           END;
```

```
2337   0   12:3   169          IF IVALID OR FEOF THEN
2338   0   12:4   175            IF NEG THEN I := -I ELSE (*NADA*)
2339   0   12:3   185          ELSE SYSCOM^.IORSLT := IBADFORMAT
2340   0   12:2   190        END;
2341   0   12:1   192 1:
2342   0   12:0   192 END (*FREADINT*) ;
2343   0   12:0   212
2344   0   12:0   212 (* STRING VARIABLE INTRINSICS *)
2345   0   12:0   212
2346   0   23:D     1 PROCEDURE SCONCAT(*VAR SRC,DEST: STRING; DESTLENG: INTEGER*);
2347   0   23:0     0 BEGIN
2348   0   23:1     0   IF LENGTH(SRC)+LENGTH(DEST) <= DESTLENG THEN
2349   0   23:2    11     BEGIN
2350   0   23:3    11       MOVELEFT(SRC[1],DEST[LENGTH(DEST)+1],LENGTH(SRC));
2351   0   23:3    24       DEST[0] := CHR(LENGTH(SRC)+LENGTH(DEST))
2352   0   23:2    33     END
2353   0   23:0    34 END (*SCONCAT*) ;
2354   0   23:0    46
2355   0   24:D     1 PROCEDURE SINSERT(*VAR SRC,DEST: STRING; DESTLENG,INSINX: INTEGER*);
2356   0   24:D     5   VAR ONRIGHT: INTEGER;
2357   0   24:0     0 BEGIN
2358   0   24:1     0   IF (INSINX > 0) AND (LENGTH(SRC) > 0) AND
2359   0   24:1     9       (LENGTH(SRC)+LENGTH(DEST) <= DESTLENG) THEN
2360   0   24:2    21     BEGIN
2361   0   24:3    21       ONRIGHT := LENGTH(DEST)-INSINX+1;
2362   0   24:3    30       IF ONRIGHT > 0 THEN
2363   0   24:4    35         BEGIN
2364   0   24:5    35           MOVERIGHT(DEST[INSINX],DEST[INSINX+LENGTH(SRC)],ONRIGHT);
2365   0   24:5    46           ONRIGHT := 0
2366   0   24:4    46         END;
2367   0   24:3    49       IF ONRIGHT = 0 THEN
2368   0   24:4    54         BEGIN
2369   0   24:5    54           MOVELEFT(SRC[1],DEST[INSINX],LENGTH(SRC));
2370   0   24:5    63           DEST[0] := CHR(LENGTH(DEST)+LENGTH(SRC))
2371   0   24:4    72         END
2372   0   24:2    73     END
2373   0   24:0    73 END (*SINSERT*) ;
2374   0   24:0    86
2375   0   25:D     1 PROCEDURE SCOPY(*VAR SRC,DEST: STRING; SRCINX,COPYLENG: INTEGER*);
2376   0   25:0     0 BEGIN DEST := '';
2377   0   25:1     6   IF (SRCINX > 0) AND (COPYLEN > 0) AND
```

```
2376   0   25:1    13        (SRCINX+COPYLENG-1 <= LENGTH(SRC)) THEN
2379   0   25:2    25      BEGIN
2380   0   25:3    25        MOVELEFT(SRC[SRCINX],DEST[1],COPYLENG);
2381   0   25:3    32        DEST[0] := CHR(COPYLENG)
2382   0   25:2    35      END
2383   0   25:0    36 END (*SCOPY*) ;
2384   0   25:0    48
2385   0   26:D     1 PROCEDURE SDELETE(*VAR DEST: STRING; DELINX,DELLENG: INTEGER*);
2386   0   26:D     4   VAR ONRIGHT: INTEGER;
2387   0   26:0     0 BEGIN
2388   0   26:1     0   IF (DELINX > 0) AND (DELLENG > 0) THEN
2389   0   26:2     9     BEGIN
2390   0   26:3     9       ONRIGHT := LENGTH(DEST)-DELINX-DELLENG+1;
2391   0   26:3    20       IF ONRIGHT = 0 THEN DEST[0] := CHR(DELINX-1)
2392   0   26:3    30       ELSE
2393   0   26:4    33         IF ONRIGHT > 0 THEN
2394   0   26:5    38           BEGIN
2395   0   26:6    38             MOVELEFT(DEST[DELINX+DELLENG],DEST[DELINX],ONRIGHT);
2396   0   26:6    47             DEST[0] := CHR(LENGTH(DEST)-DELLENG)
2397   0   26:5    54           END
2398   0   26:2    55     END
2399   0   26:0    55 END (*SDELETE*) ;
2400   0   26:0    68
2401   0   27:D     3 FUNCTION SPOS(*VAR TARGET, SRC: STRING*);
2402   0   27:D     5 LABEL 1;
2403   0   27:D     5 VAR  TEMPLOC,DIST: INTEGER;
2404   0   27:D     7      FIRSTCH: CHAR;
2405   0   27:D     8      TEMP: STRING;
2406   0   27:0     0 BEGIN SPOS := 0;
2407   0   27:1     3   IF LENGTH(TARGET) > 0 THEN
2408   0   27:2    10     BEGIN
2409   0   27:3    10       FIRSTCH := TARGET[1];
2410   0   27:3    15       TEMPLOC := 1;
2411   0   27:3    18       DIST := LENGTH(SRC)-LENGTH(TARGET) + 1;
2412   0   27:3    29       TEMP[0] :=  TARGET[0];
2413   0   27:3    36       WHILE TEMPLOC <= DIST DO
2414   0   27:4    41         BEGIN
2415   0   27:5    41           TEMPLOC := TEMPLOC + SCAN(DIST-TEMPLOC,=FIRSTCH,SRC[TEMPLOC]) ;
2416   0   27:5    55           IF TEMPLOC>DIST THEN
2417   0   27:6    60              GOTO 1;
2418   0   27:5    62           MOVELEFT(SRC[TEMPLOC],TEMP[1],LENGTH(TARGET));
```

```
2419   0   27:5    72            IF TEMP=TARGET THEN
2420   0   27:6    79              BEGIN SPOS := TEMPLOC; GOTO 1 END;
2421   0   27:5    84            TEMPLOC := TEMPLOC+1
2422   0   27:4    85          END
2423   0   27:2    89      END;
2424   0   27:1    91  1:
2425   0   27:0    91  END (*SPOS*) ;
2426   0   27:0   106
2427   0   27:0   106  (* MAIN DRIVER OF SYSTEM *)
2428   0   27:0   106
2429   0   43:D     1  PROCEDURE COMMAND;
2430   0   43:D     1    VAR T: INTEGER;
2431   0   43:0     0  BEGIN STATE := HALTINIT;
2432   0   43:1     4    REPEAT
2433   0   43:2     4      RELEASE(EMPTYHEAP);
2434   0   43:2     9      WHILE UNITABLE[SYSCOM^.SYSUNIT].UVID <> SYVID DO
2435   0   43:3    25        BEGIN
2436   0   43:4    25          PL := 'PUT IN :';
2437   0   43:4    41          INSERT(SYVID,PL,8);
2438   0   43:4    52          PROMPT; T := 4000;
2439   0   43:4    59          REPEAT T := T-1
2440   0   43:4    60          UNTIL T = 0;
2441   0   43:4    69          IF FETCHDIR(SYSCOM^.SYSUNIT) THEN
2442   0   43:3    79        END;
2443   0   43:2    81      STATE := GETCMD(STATE);
2444   0   43:2    92      CASE STATE OF
2445   0   43:2    97        UPROGNOU,UPROGUOK,SYSPROG,
2446   0   43:2    97        COMPONLY,COMPANDGO,COMPDEBUG,
2447   0   43:2    97        LINKANDGO,LINKDEBUG:
2448   0   43:3    97          USERPROGRAM(NIL,NIL);
2449   0   43:2   104        DEBUGCALL:
2450   0   43:3   104          DEBUGGER
2451   0   43:2   104      END;
2452   0   43:2   134      IF STATE IN [COMPONLY,COMPANDGO,COMPDEBUG] THEN
2453   0   43:3   144        IF USERINFO.ERRNUM = 0 THEN
2454   0   43:4   151          BEGIN    [THIS IS CONTINUED IN FINISHCOMP]
2455   0   43:5   151            FCLOSE(USERINFO.CODEFIBP^,CLOCK);
2456   0   43:5   157            IF ORD(IORESULT) <> ORD(INOERROR) THEN
2457   0   43:6   163              BEGIN
2458   0   43:7   163                T := IORESULT;
2459   0   43:7   167                WRITELN(OUTPUT);
```

```
2460    0    43:7    173                CLEARLINE;
2461    0    43:7    175                PRINTERROR(IOC[IOERROR],T);
2462    0    43:6    180                END;
2463    0    43:4    180              END;
2464    0    43:2    180        IF STATE IN [UPROGNOU,UPROGUOK] THEN
2465    0    43:3    188          BEGIN
2466    0    43:4    188            FCLOSE(GFILES[0]^,CNORMAL);
2467    0    43:4    198            FCLOSE(GFILES[1]^,CLOCK)
2468    0    43:3    206          END;
2469    0    43:2    208        IF UNITBUSY(1) OR UNITBUSY(2) THEN
2470    0    43:3    217          UNITCLEAR(1)
2471    0    43:1    218     UNTIL STATE = HALTINIT
2472    0    43:0    223 END (*COMMAND*) ;
2473    0    43:0    246
2474    0     1:0      0 BEGIN (*UCSD PASCAL SYSTEM*)
2475    0     1:1      0   EMPTYHEAP := NIL;
2476    0     1:1      3   INITIALIZE;
2477    0     1:1      6   REPEAT
2478    0     1:2      6     COMMAND;
2479    0     1:2      8     IF EMPTYHEAP <> NIL THEN
2480    0     1:3     14       INITIALIZE
2481    0     1:1     14   UNTIL EMPTYHEAP = NIL
2482    0     1:0     19 END (*PASCALSYSTEM*) .
```

```
  1    1     1:0     1 (*$L PRINTER:*)
  1    1     1:0     1 [$I GLOBALS.TEXT]
  2    1     1:0     1 (*$U-*)
  3    1     1:0     1     [$S+]
  4    1     1:0     1     (*************************************************************)
  5    1     1:0     1     (*                                                           *)
  6    1     1:D     1     (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
  7    1     1:D     1     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
  8    1     1:D     1     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
  9    1     1:D     1     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
 10    1     1:D     1     (*                                                           *)
 11    1     1:D     1     (*************************************************************)
 12    1     1:D     1
 13    0     1:D     1 PROGRAM PASCALSYSTEM;
 14    0     1:D     1
 15    0     1:D     1 (*************************************************)
 16    0     1:D     1 (*                                               *)
 17    0     1:D     1 (*    UCSD PASCAL OPERATING SYSTEM               *)
 18    0     1:D     1 (*                                               *)
 19    0     1:D     1 (*    RELEASE LEVEL:  I.3   AUGUST, 1977         *)
 20    0     1:D     1 (*                    I.4   JANUARY, 1978        *)
 21    0     1:D     1 (*                    I.5   SEPTEMBER, 1978      *)
 22    0     1:D     1 (*                    II.0  FEBRUARY, 1978 BD    *)
 23    0     1:D     1 (*                                               *)
 24    0     1:D     1 (*    WRITTEN BY ROGER T. SUMNER                 *)
 25    0     1:D     1 (*    WINTER 1977                                *)
 26    0     1:D     1 (*                                               *)
 27    0     1:D     1 (*    INSTITUTE FOR INFORMATION SYSTEMS          *)
 28    0     1:D     1 (*    UC SAN DIEGO, LA JOLLA, CA                 *)
 29    0     1:D     1 (*                                               *)
 30    0     1:D     1 (*    KENNETH L. BOWLES, DIRECTOR                *)
 31    0     1:D     1 (*                                               *)
 32    0     1:D     1 (*************************************************)
 33    0     1:D     1
 34    0     1:D     1 CONST
 35    0     1:D     1       MMAXINT = 32767;    (*MAXIMUM INTEGER VALUE*)
 36    0     1:D     1       MAXUNIT = 12;       (*MAXIMUM PHYSICAL UNIT # FOR UREAD*)
 37    0     1:D     1       MAXDIR = 77;        (*MAX NUMBER OF ENTRIES IN A DIRECTORY*)
 38    0     1:D     1       VIDLENG = 7;        (*NUMBER OF CHARS IN A VOLUME ID*)
 39    0     1:D     1       TIDLENG = 15;       (*NUMBER OF CHARS IN TITLE ID*)
 40    0     1:D     1       MAXSEG = 15;        (*MAX CODE SEGMENT NUMBER*)
```

```
41   0   1:0   1      FBLKSIZE = 512;      (*STANDARD DISK BLOCK LENGTH*)
42   0   1:0   1      DIRBLK = 2;          (*DISK ADDR OF DIRECTORY*)
43   0   1:0   1      AGELIMIT = 300;      (*MAX AGE FOR GDIRP...IN TICKS*)
44   0   1:0   1      EOL = 13;            (*END-OF-LINE...ASCII CR*)
45   0   1:0   1      DLE = 16;            (*BLANK COMPRESSION CODE*)
46   0   1:0   1      NAME_LEN = 23;       [LENGTH OF CONCAT(VIDLENG,':',TIDLENG)]
47   0   1:0   1      FILL_LEN = 11;       [MAXIMUM # OF NULLS IN FILLER]
48   0   1:0   1
49   0   1:0   1  TYPE
50   0   1:0   1
51   0   1:D   1      IORSLTWD = (INOERROR,IBADBLOCK,IBADUNIT,IBADMODE,ITIMEOUT,
52   0   1:0   1                  ILOSTUNIT,ILOSTFILE,IBADTITLE,INOROOM,INOUNIT,
53   0   1:D   1                  INOFILE,IDUPFILE,INOTCLOSED,INOTOPEN,IBADFORMAT,
54   0   1:D   1                  ISTRGOVFL);
55   0   1:D   1                                        (*COMMAND STATES...SEE GETCMD*)
56   0   1:D   1
57   0   1:D   1
58   0   1:D   1      CMDSTATE = (HALTINIT,DEBUGCALL,
59   0   1:D   1                  UPROGNOU,UPROGUOK,SYSPROG,
60   0   1:D   1                  COMPONLY,COMPANDGO,COMPDEBUG,
61   0   1:D   1                  LINKANDGO,LINKDEBUG);
62   0   1:D   1                                        (*CODE FILES USED IN GETCMD*)
63   0   1:D   1
64   0   1:D   1
65   0   1:D   1      SYSFILE = (ASSMBLER,COMPILER,EDITOR,FILER,LINKER);
66   0   1:D   1                                        (*ARCHIVAL INFO...THE DATE*)
67   0   1:D   1
68   0   1:D   1
69   0   1:D   1      DATEREC = PACKED RECORD
70   0   1:D   1                  MONTH: 0..12;        (*0 IMPLIES DATE NOT MEANINGFUL*)
71   0   1:D   1                  DAY: 0..31;          (*DAY OF MONTH*)
72   0   1:D   1                  YEAR: 0..100         (*100 IS TEMP DISK FLAG*)
73   0   1:D   1                END (*DATEREC*) ;
74   0   1:D   1                                        (*VOLUME TABLES*)
75   0   1:D   1
76   0   1:D   1      UNITNUM = 0..MAXUNIT;
77   0   1:D   1      VID = STRING[VIDLENG];
78   0   1:D   1                                        (*DISK DIRECTORIES*)
79   0   1:D   1
80   0   1:D   1      DIRRANGE = 0..MAXDIR;
81   0   1:D   1      TID = STRING[TIDLENG];
```

```
 82   0    1:0     1      FULL_ID = STRING[NAME_LEN];
 83   0    1:0     1
 84   0    1:0     1      FILE_TABLE = ARRAY [SYSFILE] OF FULL_ID;
 85   0    1:0     1
 86   0    1:0     1      FILEKIND = (UNTYPEDFILE,XDSKFILE,CODEFILE,TEXTFILE,
 87   0    1:0     1                  INFOFILE,DATAFILE,GRAFFILE,FOTOFILE,SECUREDIR);
 88   0    1:0     1
 89   0    1:0     1      DIRENTRY = PACKED RECORD
 90   0    1:0     1                   DFIRSTBLK: INTEGER;    (*FIRST PHYSICAL DISK ADDR*)
 91   0    1:0     1                   DLASTBLK: INTEGER;     (*POINTS AT BLOCK FOLLOWING*)
 92   0    1:0     1                   CASE DFKIND: FILEKIND OF
 93   0    1:0     1                     SECUREDIR,
 94   0    1:0     1                     UNTYPEDFILE: (*ONLY IN DIR[0]...VOLUME INFO*)
 95   0    1:0     1                        (FILLER1 : 0..2048; [FOR DOWNWARD COMPATIBILITY,13 BITS]
 96   0    1:0     1                         DVID: VID;                (*NAME OF DISK VOLUME*)
 97   0    1:0     1                         DEOVBLK: INTEGER;         (*LASTBLK OF VOLUME*)
 98   0    1:0     1                         DNUMFILES: DIRRANGE;      (*NUM FILES IN DIR*)
 99   0    1:0     1                         DLOADTIME: INTEGER;       (*TIME OF LAST ACCESS*)
100   0    1:0     1                         DLASTBOOT: DATEREC);      (*MOST RECENT DATE SETTING*)
101   0    1:0     1                     XDSKFILE,CODEFILE,TEXTFILE,INFOFILE,
102   0    1:0     1                     DATAFILE,GRAFFILE,FOTOFILE:
103   0    1:0     1                        (FILLER2 : 0..1024; [FOR DOWNWARD COMPATIBILITY]
104   0    1:0     1                         STATUS : BOOLEAN;          [FOR FILER WILDCARDS]
105   0    1:0     1                         DTID: TID;                (*TITLE OF FILE*)
106   0    1:0     1                         DLASTBYTE: 1..FBLKSIZE;   (*NUM BYTES IN LAST BLOCK*)
107   0    1:0     1                         DACCESS: DATEREC)         (*LAST MODIFICATION DATE*)
108   0    1:0     1                END (*DIRENTRY*) ;
109   0    1:0     1
110   0    1:0     1      DIRP = ^DIRECTORY;
111   0    1:0     1
112   0    1:0     1      DIRECTORY = ARRAY [DIRRANGE] OF DIRENTRY;
113   0    1:0     1
114   0    1:0     1                                            (*FILE INFORMATION*)
115   0    1:0     1
116   0    1:0     1      CLOSETYPE = (CNORMAL,CLOCK,CPURGE,CCRUNCH);
117   0    1:0     1      WINDOWP = ^WINDOW;
118   0    1:0     1      WINDOW = PACKED ARRAY [0..0] OF CHAR;
119   0    1:0     1      FIBP = ^FIB;
120   0    1:0     1
121   0    1:0     1      FIB = RECORD
122   0    1:0     1              FWINDOW: WINDOWP;  (*USER WINDOW...F^, USED BY GET-PUT*)
```

```
123   0    1:0    1            FEOF,FEOLN: BOOLEAN;
124   0    1:0    1            FSTATE: (FJANDW,FNEEDCHAR,FGOTCHAR);
125   0    1:0    1            FRECSIZE: INTEGER; (*IN BYTES...0=>BLOCKFILE, 1=>CHARFILE*)
126   0    1:0    1            CASE FISOPEN: BOOLEAN OF
127   0    1:0    1              TRUE: (FISBLKD: BOOLEAN; (*FILE IS ON BLOCK DEVICE*)
128   0    1:0    1                     FUNIT: UNITNUM;   (*PHYSICAL UNIT #*)
129   0    1:0    1                     FVID: VID;        (*VOLUME NAME*)
130   0    1:0    1                     FREPTCNT,         (* # TIMES F^ VALID W/O GET*)
131   0    1:0    1                     FNXTBLK,          (*NEXT REL BLOCK TO IO*)
132   0    1:0    1                     FMAXBLK: INTEGER; (*MAX REL BLOCK ACCESSED*)
133   0    1:0    1                     FMODIFIED:BOOLEAN;(*PLEASE SET NEW DATE IN CLOSE*)
134   0    1:0    1                     FHEADER: DIRENTRY;(*COPY OF DISK DIR ENTRY*)
135   0    1:0    1                     CASE FSOFTBUF: BOOLEAN OF (*DISK GET-PUT STUFF*)
136   0    1:0    1                       TRUE: (FNXTBYTE,FMAXBYTE: INTEGER;
137   0    1:0    1                              FBUFCHNGD: BOOLEAN;
138   0    1:0    1                              FBUFFER: PACKED ARRAY [0..FBLKSIZE] OF CHAR))
139   0    1:0    1          END (*FIB*) ;
140   0    1:0    1
141   0    1:0    1                                          (*USER WORKFILE STUFF*)
142   0    1:0    1
143   0    1:0    1      INFOREC = RECORD
144   0    1:0    1                  SYMFIBP,CODEFIBP: FIBP;          (*WORKFILES FOR SCRATCH*)
145   0    1:0    1                  ERRSYM,ERRBLK,ERRNUM: INTEGER;   (*ERROR STUFF IN EDIT*)
146   0    1:0    1                  SLOWTERM,STUPID: BOOLEAN;        (*STUDENT PROGRAMMER ID!!*)
147   0    1:0    1                  ALTMODE: CHAR;                   (*WASHOUT CHAR FOR COMPILER*)
148   0    1:0    1                  GOTSYM,GOTCODE: BOOLEAN;         (*TITLES ARE MEANINGFUL*)
149   0    1:0    1                  WORKVID,SYMVID,CODEVID: VID;     (*PERM&CUR WORKFILE VOLUMES*)
150   0    1:0    1                  WORKTID,SYMTID,CODETID: TID      (*PERM&CUR WORKFILES TITLE*)
151   0    1:0    1                END (*INFOREC*) ;
152   0    1:0    1
153   0    1:0    1                                          (*CODE SEGMENT LAYOUTS*)
154   0    1:0    1
155   0    1:0    1      SEGRANGE = 0..MAXSEG;
156   0    1:0    1      SEGDESC = RECORD
157   0    1:0    1                  DISKADDR: INTEGER;       (*REL BLK IN CODE...ABS IN SYSCOM^*)
158   0    1:0    1                  CODELENG: INTEGER        (*# BYTES TO READ IN*)
159   0    1:0    1                END (*SEGDESC*) ;
160   0    1:0    1
161   0    1:0    1                                          (*DEBUGGER STUFF*)
162   0    1:0    1
163   0    1:0    1      BYTERANGE = 0..255;
```

```
164   0    1:D     1      TRICKARRAY = RECORD          [MEMORY DIDDLING FOR EXECERROR]
165   0    1:D     1                     CASE BOOLEAN OF
166   0    1:D     1                       TRUE : (WORD : ARRAY [0..0] OF INTEGER);
167   0    1:D     1                       FALSE : (BYTE : PACKED ARRAY [0..0] OF BYTERANGE)
168   0    1:D     1                     END;
169   0    1:D     1      MSCWP = ^ MSCW;              (*MARK STACK RECORD POINTER*)
170   0    1:D     1      MSCW = RECORD
171   0    1:D     1               STATLINK: MSCWP;   (*POINTER TO PARENT MSCW*)
172   0    1:D     1               DYNLINK: MSCWP;    (*POINTER TO CALLER'S MSCW*)
173   0    1:D     1               MSSEG,MSJTAB: ^TRICKARRAY;
174   0    1:D     1               MSIPC: INTEGER;
175   0    1:D     1               LOCALDATA: TRICKARRAY
176   0    1:D     1             END (*MSCW*) ;
177   0    1:D     1
178   0    1:D     1                                          (*SYSTEM COMMUNICATION AREA*)
179   0    1:D     1                                          (*SEE INTERPRETERS...NOTE  *)
180   0    1:D     1                                          (*THAT WE ASSUME BACKWARD  *)
181   0    1:D     1                                          (*FIELD ALLOCATION IS DONE *)
182   0    1:D     1      SYSCOMREC = RECORD
183   0    1:D     1                    IORSLT: IORSLTWD;     (*RESULT OF LAST IO CALL*)
184   0    1:D     1                    XEQERR: INTEGER;      (*REASON FOR EXECERROR CALL*)
185   0    1:D     1                    SYSUNIT: UNITNUM;     (*PHYSICAL UNIT OF BOOTLOAD*)
186   0    1:D     1                    BUGSTATE: INTEGER;    (*DEBUGGER INFO*)
187   0    1:D     1                    GDIRP: DIRP;          (*GLOBAL DIR POINTER,SEE VOLSEARCH*)
188   0    1:D     1                    LASTMP,STKBASE,BOMBP: MSCWP;
189   0    1:D     1                    MEMTOP,SEG,JTAB: INTEGER;
190   0    1:D     1                    BOMBIPC: INTEGER;     (*WHERE XEQERR BLOWUP WAS*)
191   0    1:D     1                    HLTLINE: INTEGER;     (*MORE DEBUGGER STUFF*)
192   0    1:D     1                    BRKPTS: ARRAY [0..3] OF INTEGER;
193   0    1:D     1                    RETRIES: INTEGER;     (*DRIVERS PUT RETRY COUNTS*)
194   0    1:D     1                    EXPANSION: ARRAY [0..8] OF INTEGER;
195   0    1:D     1                    HIGHTIME,LOWTIME: INTEGER;
196   0    1:D     1                    MISCINFO: PACKED RECORD
197   0    1:D     1                                NOBREAK,STUPID,SLOWTERM,
198   0    1:D     1                                HASXYCRT,HASLCCRT,HAS8510A,HASCLOCK: BOOLEAN;
199   0    1:D     1                                USERKIND:(NORMAL, AQUIZ, BOOKER, PQUIZ);
200   0    1:D     1                                IS_FLIPT : BOOLEAN
201   0    1:D     1                              END;
202   0    1:D     1                    CRTTYPE: INTEGER;
203   0    1:D     1                    CRTCTRL: PACKED RECORD
204   0    1:D     1                                 RLF,NDFS,ERASEEOL,ERASEEOS,HOME,ESCAPE: CHAR;
```

```
205   0    1:0     1                              BACKSPACE: CHAR;
206   0    1:0     1                              FILLCOUNT: 0..255;
207   0    1:0     1                              CLEARSCREEN, CLEARLINE: CHAR;
208   0    1:0     1                              PREFIXED: PACKED ARRAY [0..8] OF BOOLEAN
209   0    1:0     1                            END;
210   0    1:0     1                      CRTINFO: PACKED RECORD
211   0    1:0     1                              WIDTH,HEIGHT: INTEGER;
212   0    1:0     1                              RIGHT,LEFT,DOWN,UP: CHAR;
213   0    1:0     1                              BADCH,CHARDEL,STOP,BREAK,FLUSH,EOF: CHAR;
214   0    1:0     1                              ALTMODE,LINEDEL: CHAR;
215   0    1:0     1                              BACKSPACE,ETX,PREFIX: CHAR;
216   0    1:0     1                              PREFIXED: PACKED ARRAY [0..13] OF BOOLEAN
217   0    1:0     1                            END;
218   0    1:0     1                      SEGTABLE: ARRAY [SEGRANGE] OF
219   0    1:0     1                                  RECORD
220   0    1:0     1                                    CODEUNIT: UNITNUM;
221   0    1:0     1                                    CODEDESC: SEGDESC
222   0    1:0     1                                  END
223   0    1:0     1                  END (*SYSCOM*);
224   0    1:0     1
225   0    1:0     1      MISCINFOREC = RECORD
226   0    1:0     1                      MSYSCOM: SYSCOMREC
227   0    1:0     1                    END;
228   0    1:0     1
229   0    1:0     1  VAR
230   0    1:0     1      SYSCOM: ^SYSCOMREC;                    (*MAGIC PARAM...SET UP IN BOOT*)
231   0    1:0     2      GFILES: ARRAY [0..5] OF FIBP;          (*GLOBAL FILES, 0=INPUT, 1=OUTPUT*)
232   0    1:0     8      USERINFO: INFOREC;                     (*WORK STUFF FOR COMPILER ETC*)
233   0    1:0    54      EMPTYHEAP: ^INTEGER;                   (*HEAP MARK FOR MEM MANAGING*)
234   0    1:0    55      INPUTFIB,OUTPUTFIB,                    (*CONSOLE FILES...GFILES ARE COPIES*)
235   0    1:0    55      SYSTERM,SWAPFIB: FIBP;                 (*CONTROL AND SWAPSPACE FILES*)
236   0    1:0    59      SYVID,DKVID: VID;                      (*SYSUNIT VOLID & DEFAULT VOLID*)
237   0    1:0    67      THEDATE: DATEREC;                      (*TODAY...SET IN FILER OR SIGN ON*)
238   0    1:0    68      DEBUGINFO: ^INTEGER;                   (*DEBUGGERS GLOBAL INFO WHILE RUNIN*)
239   0    1:0    69      STATE: CMDSTATE;                       (*FOR GETCOMMAND*)
240   0    1:0    70      PL: STRING;                            (*PROMPTLINE STRING...SEE PROMPT*)
241   0    1:0   111      IPOT: ARRAY [0..4] OF INTEGER;         (*INTEGER POWERS OF TEN*)
242   0    1:0   116      FILLER: STRING[FILL_LEN];              (*NULLS FOR CARRIAGE DELAY*)
243   0    1:0   122      DIGITS: SET OF '0'..'9';
244   0    1:0   126      UNITABLE: ARRAY [UNITNUM] OF (*0 NOT USED*)
245   0    1:0   126                  RECORD
```

```
246   0    1:D   126                    UVID: VID;     (*VOLUME ID FOR UNIT*)
247   0    1:D   126                    CASE UISBLKD: BOOLEAN OF
248   0    1:D   126                      TRUE: (UEOVBLK: INTEGER)
249   0    1:D   126                  END (*UNITABLE*) ;
250   0    1:D   204      FILENAME : FILE_TABLE;
251   0    1:D   264
252   0    1:D   264 (*-----------------------------------------------------------------------*)
253   0    1:D   264 (* SYSTEM PROCEDURE FORWARD DECLARATIONS *)
254   0    1:D   264 (* THESE ARE ADDRESSED BY OBJECT CODE... *)
255   0    1:D   264 (*  DO NOT MOVE WITHOUT CAREFUL THOUGHT  *)
256   0    1:D   264
257   0    2:D     1 PROCEDURE EXECERROR;
258   0    2:D     1   FORWARD;
259   0    3:D     1 PROCEDURE FINIT(VAR F: FIB; WINDOW: WINDOWP; RECWORDS: INTEGER);
260   0    3:D     4   FORWARD;
261   0    4:D     1 PROCEDURE FRESET(VAR F: FIB);
262   0    4:D     2   FORWARD;
263   0    5:D     1 PROCEDURE FOPEN(VAR F: FIB; VAR FTITLE: STRING;
264   0    5:D     3                   FOPENOLD: BOOLEAN; JUNK: FIBP);
265   0    5:D     5   FORWARD;
266   0    6:D     1 PROCEDURE FCLOSE(VAR F: FIB; FTYPE: CLOSETYPE);
267   0    6:D     3   FORWARD;
268   0    7:D     1 PROCEDURE FGET(VAR F: FIB);
269   0    7:D     2   FORWARD;
270   0    8:D     1 PROCEDURE FPUT(VAR F: FIB);
271   0    8:D     2   FORWARD;
272   0    9:D     1 PROCEDURE XSEEK;
273   0    9:D     1   FORWARD;
274   0   10:D     3 FUNCTION FEOF(VAR F: FIB): BOOLEAN;
275   0   10:D     4   FORWARD;
276   0   11:D     3 FUNCTION FEOLN(VAR F: FIB): BOOLEAN;
277   0   11:D     4   FORWARD;
278   0   12:D     1 PROCEDURE FREADINT(VAR F: FIB; VAR I: INTEGER);
279   0   12:D     3   FORWARD;
280   0   13:D     1 PROCEDURE FWRITEINT(VAR F: FIB; I,RLENG: INTEGER);
281   0   13:D     4   FORWARD;
282   0   14:D     1 PROCEDURE XREADREAL;
283   0   14:D     1   FORWARD;
284   0   15:D     1 PROCEDURE XWRITEREAL;
285   0   15:D     1   FORWARD;
286   0   16:D     1 PROCEDURE FREADCHAR(VAR F: FIB; VAR CH: CHAR);
```

```
287   0   16:D     3   FORWARD;
288   0   17:D     1 PROCEDURE FWRITECHAR(VAR F: FIB; CH: CHAR; RLENG: INTEGER);
289   0   17:D     4   FORWARD;
290   0   18:D     1 PROCEDURE FREADSTRING(VAR F: FIB; VAR S: STRING; SLENG: INTEGER);
291   0   18:D     4   FORWARD;
292   0   19:D     1 PROCEDURE FWRITESTRING(VAR F: FIB; VAR S: STRING; RLENG: INTEGER);
293   0   19:D     4   FORWARD;
294   0   20:D     1 PROCEDURE FWRITEBYTES(VAR F: FIB; VAR A: WINDOW; RLENG,ALENG: INTEGER);
295   0   20:D     5   FORWARD;
296   0   21:D     1 PROCEDURE FREADLN(VAR F: FIB);
297   0   21:D     2   FORWARD;
298   0   22:D     1 PROCEDURE FWRITELN(VAR F: FIB);
299   0   22:D     2   FORWARD;
300   0   23:D     1 PROCEDURE SCONCAT(VAR DEST,SRC: STRING; DESTLENG: INTEGER);
301   0   23:D     4   FORWARD;
302   0   24:D     1 PROCEDURE SINSERT(VAR SRC,DEST: STRING; DESTLENG,INSINX: INTEGER);
303   0   24:D     5   FORWARD;
304   0   25:D     1 PROCEDURE SCOPY(VAR SRC,DEST: STRING; SRCINX,COPYLENG: INTEGER);
305   0   25:D     5   FORWARD;
306   0   26:D     1 PROCEDURE SDELETE(VAR DEST: STRING; DELINX,DELLENG: INTEGER);
307   0   26:D     4   FORWARD;
308   0   27:D     3 FUNCTION SPOS(VAR TARGET,SRC: STRING): INTEGER;
309   0   27:D     5   FORWARD;
310   0   28:D     3 FUNCTION FBLOCKIO(VAR F: FIB; VAR A: WINDOW; I: INTEGER;
311   0   28:D     6                   NBLOCKS,RBLOCK: INTEGER; DOREAD: BOOLEAN): INTEGER;
312   0   28:D     9   FORWARD;
313   0   29:D     1 PROCEDURE FGOTOXY(X,Y: INTEGER);
314   0   29:D     3   FORWARD;
315   0   29:D     3
316   0   29:D     3 (* NON FIXED FORWARD DECLARATIONS *)
317   0   29:D     3
318   0   30:D     3 FUNCTION VOLSEARCH(VAR FVID: VID; LOOKHARD: BOOLEAN;
319   0   30:D     5                    VAR FDIR: DIRP): UNITNUM;
320   0   30:D     6   FORWARD;
321   0   31:D     1 PROCEDURE WRITEDIR(FUNIT: UNITNUM; FDIR: DIRP);
322   0   31:D     3   FORWARD;
323   0   32:D     3 FUNCTION DIRSEARCH(VAR FTID: TID; FINDPERM: BOOLEAN; FDIR: DIRP): DIRRANGE;
324   0   32:D     6   FORWARD;
325   0   33:D     3 FUNCTION SCANTITLE(FTITLE: STRING; VAR FVID: VID; VAR FTID: TID;
326   0   33:D     6                    VAR FSEGS: INTEGER; VAR FKIND: FILEKIND): BOOLEAN;
327   0   33:D    49   FORWARD;
```

```
328   0    34:0     1 PROCEDURE DELENTRY(FINX: DIRRANGE; FDIR: DIRP);
329   0    34:0     3   FORWARD;
330   0    35:0     1 PROCEDURE INSENTRY(VAR FENTRY: DIRENTRY; FINX: DIRRANGE; FDIR: DIRP);
331   0    35:0     4   FORWARD;
332   0    36:0     1 PROCEDURE HOMECURSOR;
333   0    36:0     1   FORWARD;
334   0    37:0     1 PROCEDURE CLEARSCREEN;
335   0    37:0     1   FORWARD;
336   0    38:0     1 PROCEDURE CLEARLINE;
337   0    38:0     1   FORWARD;
338   0    39:0     1 PROCEDURE PROMPT;
339   0    39:0     1   FORWARD;
340   0    40:0     3 FUNCTION SPACEWAIT(FLUSH: BOOLEAN): BOOLEAN;
341   0    40:0     4   FORWARD;
342   0    41:0     3 FUNCTION GETCHAR(FLUSH: BOOLEAN): CHAR;
343   0    41:0     4   FORWARD;
344   0    42:0     3 FUNCTION FETCHDIR(FUNIT:UNITNUM) : BOOLEAN;
345   0    42:0     4   FORWARD;
346   0    43:0     1 PROCEDURE COMMAND;
347   0    43:0     1   FORWARD;
348   0    43:0     1
349   0    43:0     1
350   0    43:0     1 [$I GLOBALS.TEXT]
350   0    43:0     1 [$I FILER.VARS.TEXT]
351   0    43:0     1 [*********************************************]
352   0    43:0     1 [                                             ]
353   0    43:0     1 [            UCSD PASCAL FILEHANDLER          ]
354   0    43:0     1 [                                             ]
355   0    43:0     1 [    RELEASE LEVEL:  II.0 FEBRUARY, 1979      ]
356   0    43:0     1 [                                             ]
357   0    43:0     1 [                                             ]
358   0    43:0     1 [    WRITTEN BY ROGER T. SUMNER               ]
359   0    43:0     1 [      RELEASE LEVEL I.4, WINTER 1977         ]
360   0    43:0     1 [                                             ]
361   0    43:0     1 [    WRITTEN BY STEVEN S THOMSON              ]
362   0    43:0     1 [      RELEASE LEVEL F.5A SUMMER 1979         ]
363   0    43:0     1 [      RELEASE LEVEL II.0 WINTER 1978-79      ]
364   0    43:0     1 [                                             ]
365   0    43:0     1 [    INSTITUTE FOR INFORMATION SYSTSEMS       ]
366   0    43:0     1 [    UC SAN DIEGO, LA JOLLA, CALIFORNIA       ]
367   0    43:0     1 [                                             ]
```

```
368   0   43:D    1 [    KENNETH L. BOWLES, DIRECTOR                          ]
369   0   43:D    1 [                                                         ]
370   0   43:D    1 [                                                         ]
371   0   43:D    1 [*********************************************************]
372   0   43:D    1
373   0   43:D    1
374   0   43:D    1 [     COPYRIGHT (C) 1979 REGENTS OF THE UNIVERSITY OF CALIFORNIA.      ]
375   0   43:D    1 [     PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-       ]
376   0   43:D    1 [     TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE      ]
377   0   43:D    1 [     OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.            ]
378   0   43:D    1
379   0   43:D    1
380   1    1:D    1 SEGMENT PROCEDURE FILEHANDLER(ZZZZZ,ZZZZZZ : INTEGER);
381   1    1:D    3
382   1    1:D    3 CONST
383   1    1:D    3   DIRLASTBLK     = 6;      DUPDIRLASTBLK = 10;     SHSTRLENG = 25;
384   1    1:D    3   MAXTITLE       = 40;     HALFMAXDIR    = 39;
385   1    1:D    3
386   1    1:D    3   VOLONLINE      = 1013;   FILEUNBLKDEXP = 1020;   NOWRK          = 1027;
387   1    1:D    3   TEXTLOST       = 1014;   FILEBLKDEXP   = 1021;   NOWILD         = 1028;
388   1    1:D    3   CODELOST       = 1015;   FILEVOLEXP    = 1022;   BADFORM        = 1029;
389   1    1:D    3   FOUNDFILE      = 1016;   VOLEXP        = 1023;   ILLFILEVOL     = 1030;
390   1    1:D    3   BLKDEXP        = 1017;   FILEFULL      = 1024;   ILLCHANGE      = 1031;
391   1    1:D    3   UNBLKDEXP      = 1018;   WRKSAVED      = 1025;   BADDEST        = 1032;
392   1    1:D    3   FILEEXP        = 1019;   NODIR         = 1026;   BLKD           = 1033;
393   1    1:D    3                                                    UNBLKD         = 1034;
394   1    1:D    3
395   1    1:D    3 TYPE
396   1    1:D    3   UNTYPED  = FILE;
397   1    1:D    3
398   1    1:D    3   TIDRANGE = 0..TIDLENG;
399   1    1:D    3
400   1    1:D    3   MATCHES  = (FILEFOUND, NOFILES, FILESNOGOOD, ABORTIT);
401   1    1:D    3   LOCATION = (SOURCE,DESTINATION,NEITHER);
402   1    1:D    3   CHECKS   = (BADTITLE, BADUNIT, NOVOL, BADDIR,
403   1    1:D    3                 BADFILE, UNBLKDVOL, OKDIR, OKFILE);
404   1    1:D    3
405   1    1:D    3   CHCKS    = SET OF CHECKS;
406   1    1:D    3
407   1    1:D    3   LONGSTRING  = STRING[255];
408   1    1:D    3   SHORTSTRING = STRING[SHSTRLENG];
```

```
409   1    1:D     3   STRNG        = STRING[MAXTITLE];
410   1    1:D     3
411   1    1:D     3   ABLOCK       = ARRAY [0..255] OF INTEGER;
412   1    1:D     3
413   1    1:D     3   BITMAP       = PACKED RECORD
414   1    1:D     3                    DIRENTRY : PACKED ARRAY [DIRRANGE] OF BOOLEAN;
415   1    1:D     3                    ENTRIES : DIRRANGE;
416   1    1:D     3                  END;
417   1    1:D     3
418   1    1:D     3
419   1    1:D     3 VAR
420   1    1:D     3   GBUFBLKS                  : INTEGER;      [ BLOCKS AVAILABLE IN TRANSFER BUFFER   ]
421   1    1:D     4   SOURCEUNIT, DESTUNIT,                     [ UNITS RELATED TO SOURCE & DEST. FILES ]
422   1    1:D     4   GUNIT                                     [ UNIT # THAT LAST VOLSEARCH RETURNED   ]
423   1    1:D     4                              : UNITNUM;
424   1    1:D     7
425   1    1:D     7   CH                         : CHAR;         [ GENERAL PURPOSE CHARARCTER            ]
426   1    1:D     8
427   1    1:D     8   GDIR                       : DIRP;         [ POINTER TO THE DIRECTORY IN USE       ]
428   1    1:D     9   LFIBP                      : FIBP;         [ POINTER TO THE HEADER OF FILE LFIB    ]
429   1    1:D    10
430   1    1:D    10   FAST,                                     [ SYSCOM^ [NOT SLOWTERM & (WIDTH > 79)] ]
431   1    1:D    10   MARKING,                                  [ MUST USE STATUS BIT IN DIRECTORY      ]
432   1    1:D    10   QUESTION, WILDCARD,                       [ IS WILDCARD OPTION BEING USED ?       ]
433   1    1:D    10   TEXTSAVED, CODESAVED                      [ WORKFILES SAVED ?                     ]
434   1    1:D    10                              : BOOLEAN;
435   1    1:D    16
436   1    1:D    16   LASTSTATE                  : CHECKS;        [ STATE OF LAST CALL TO SCANINPUT      ]
437   1    1:D    17   FOUND                      : MATCHES;       [ RESULT OF DIR. SEARCH FOR A FILE     ]
438   1    1:D    18
439   1    1:D    18   GBUF                       : WINDOWP;       [ POINTER TO THE TRANSFER BUFFER       ]
440   1    1:D    19
441   1    1:D    19   GKIND                      : FILEKIND;      [ FILETYPE (E.G., TEXT,CODE,DATA...)   ]
442   1    1:D    20
443   1    1:D    20   DIRMAP                     : BITMAP;        [ KEEPS TRACK OF THE FILES TO BE USED  ]
444   1    1:D    26                                              [ IN A WILDCARD OPERATION              ]
445   1    1:D    26
446   1    1:D    26   BLOCKPTR                   : ^ABLOCK;       [ POINTER TO ONE-BLOCK OF DATA         ]
447   1    1:D    27
448   1    1:D    27   LFIB                       : UNTYPED;       [ GENERAL PURPOSE FILE                 ]
449   1    1:D    67
```

```
450   1    1:D    67   VOLNAME1, VOLNAME2,                          [ VOLUME NAMES OF SOURCE & DESTINATION ]
451   1    1:D    67                                                [ FILES RESPECTIVELY, AS INPUTTED      ]
452   1    1:D    67   DESTVID, SOURCEVID,                          [ EXPLICIT VOLUME NAME ASSOCIATED WITH ]
453   1    1:D    67                                                [ SOURCE & DEST UNITS RESPECTIVELY     ]
454   1    1:D    67   GVID,                                        [ LAST VOLNAME RETURNED BY SCANIPUT    ]
455   1    1:D    67   GVID2                                        [ LAST VOLNAME ENTERED INTO SCANIPUT   ]
456   1    1:D    67                          : VID;
457   1    1:D    91
458   1    1:D    91   SOURCETITLE,                                 [ SOURCE FILE WITH EXLICIT VOLUME NAME ]
459   1    1:D    91   STRING2,STRING4,                             [ SUFFIX STRINGS TO WILDCARDS          ]
460   1    1:D    91   GTID                                         [ LAST TITLE RETURNED BY SCANIPUT      ]
461   1    1:D    91                          : TID;
462   1    1:D   123
463   1    1:D   123   STRING1, STRING3      : SHORTSTRING;    [ PREFIX STRINGS TO WILDCARDS              ]
464   1    1:D   149   MONTHSTR              : STRING[48];     [ CONTAINS ABBR. FOR THE MONTHS            ]
465   1    1:D   174   TYPESTR               : STRING[32];     [ CONTAINS ABBR. FOR THE FILE TYPES        ]
466   1    1:D   191   FROMWHERE, TOWHERE    : STRNG;          [ SOURCE & DESTINATION FILES               ]
467   1    1:D   233   INSTRING              : LONGSTRING;     [ INPUT STRING                             ]
468   1    1:D   361
469   1    1:D   361
470   1    1:D   361 [$I FILER.VARS.TEXT]
470   1    1:D   361 [$I FILER.A.TEXT]
471   1    1:D   361 [     COPYRIGHT (C) 1979 REGENTS OF THE UNIVERSITY OF CALIFORNIA.      ]
472   1    1:D   361 [     PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-       ]
473   1    1:D   361 [     TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE      ]
474   1    1:D   361 [     OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.            ]
475   1    1:D   361
476   1    1:D   361 [*****************************************************************************]
477   1    1:D   361 [*****************************************************************************]
478   1    1:D   361
479   1    1:D   361   [ THIS PROCDURE IS CALLED IN AN INFINITE LOOP. USED TO EXIT FROM WHEN AN      ]
480   1    1:D   361   [ ERROR CONDITION IS ENCOUNTERED. WILL RETURN TO MAIN FILER PROMPT LINE       ]
481   1    2:D     1   PROCEDURE CALLPROC;
482   1    2:D     1     FORWARD;
483   1    2:D     1
484   1    2:D     1   [ TRICK PROCEDURE USED TO CHANGE A POINTER TO A UNTYPED FILE TO A POINTER TO ]
485   1    2:D     1   [ THE HEADER OF THAT FILE FIBP                                               ]
486   1    3:D     3   FUNCTION GETPTR(VAR DUMMY : UNTYPED) : FIBP;  [ DUMMY IS PLACED ON THE STACK ]
487   1    3:D     4   VAR
488   1    3:D     4     TRIX : ARRAY [0..0] OF FIBP;                [ TRIX IS PLACED ON THE STACK  ]
489   1    3:0     0   BEGIN
```

```
490   1     3:1      0     GETPTR := TRIX[-1]                                    [ WE ACCESS DUMMY AS TYPE FIBP ]
491   1     3:0      5   END;
492   1     3:0     22
493   1     3:0     22 [*************************************************************************************]
494   1     3:0     22 [*************************************************************************************]
495   1     3:0     22
496   1     3:0     22   [ INITIALIZES GLOBAL VARIABLES FOR THE FILER ]
497   1     4:0      1   PROCEDURE INITGLOBALS;
498   1     4:0      0   BEGIN
499   1     4:1      0     GVID := '';
500   1     4:1      7     STRING1 := '';
501   1     4:1     15     STRING2 := '';
502   1     4:1     22     STRING3 := '';
503   1     4:1     29     STRING4 := '';
504   1     4:1     36     TOWHERE := '';
505   1     4:1     44     VOLNAME1 := '';
506   1     4:1     51     VOLNAME2 := '';
507   1     4:1     58     FROMWHERE := '';
508   1     4:1     66     SOURCEVID := '';
509   1     4:1     73     DESTVID := '';
510   1     4:1     80     SOURCEUNIT := 0;
511   1     4:1     83     DESTUNIT := 0;
512   1     4:1     86     FOUND := NOFILES;
513   1     4:1     89     WILDCARD := FALSE;
514   1     4:1     92     QUESTION := FALSE;
515   1     4:1     95     FILLCHAR(DIRMAP,SIZEOF(DIRMAP),0)
516   1     4:0    102   END;
517   1     4:0    114
518   1     4:0    114 [------------------------- FILER ERROR MESSAGES ----------------------------]
519   1     4:0    114
520   1     4:0    114   [ WRITES OUT MOST FILER RELATED AND I/O ERRORS. IF NUMBER <> 0 AND EXXIT ]
521   1     4:0    114   [ THEN THIS PROCEDURE WILL RETURN TO THE FILER PROMPT LINE               ]
522   1     5:0      1   PROCEDURE MESSAGES(NUMBER : INTEGER; EXXIT : BOOLEAN);
523   1     5:0      3   VAR
524   1     5:0      3     STR : STRING[40];
525   1     5:0      0   BEGIN
526   1     5:1      0     STR := '';
527   1     5:1      7
528   1     5:1      7     [----------------- I/O ERRORS -------------------]
529   1     5:1      7     CASE NUMBER OF
530   1     5:1     10       1 : STR := 'PARITY (CRC) ERROR';
```

```
531   1    5:1    37        2 : STR := 'BAD UNIT NUMBER';
532   1    5:1    61        3 : STR := 'BAD I/O OPERATION';
533   1    5:1    87        4 : STR := 'TIMEOUT ERROR';
534   1    5:1   109        5 : STR := 'VOL WENT OFF-LINE';
535   1    5:1   135        6 : STR := 'FILE LOST IN DIR';
536   1    5:1   160        7 : STR := 'BAD FILE NAME';
537   1    5:1   182        8 : STR := 'NO ROOM ON VOL';
538   1    5:1   205        9 : STR := 'NO SUCH VOL ON-LINE';
539   1    5:1   233       10: STR := 'FILE NOT FOUND';
540   1    5:1   256      END;
541   1    5:1   284
542   1    5:1   284      [------------ FILER RELATED ERRORS ---------------]
543   1    5:1   284      CASE NUMBER OF
544   1    5:1   287        1013: STR := 'VOL ALREADY ON-LINE';
545   1    5:1   315        1014: STR := 'TEXT FILE LOST';
546   1    5:1   338        1015: STR := 'CODE FILE LOST';
547   1    5:1   361        1016: STR := 'FILE FOUND';
548   1    5:1   380        1017,1033: STR := 'BLKD VOL';
549   1    5:1   397        1018,1034: STR := 'UBLKD VOL';
550   1    5:1   415        1019: STR := 'FILE NAME';
551   1    5:1   433        1020: STR := 'FILE/(UNBLKD VOL)';
552   1    5:1   459        1021: STR := 'FILE/(BLKD VOL)';
553   1    5:1   483        1022: STR := 'FILE/VOL';
554   1    5:1   500        1023: STR := 'VOL NAME';
555   1    5:1   517        1024: STR := 'OUTPUT FILE FULL';
556   1    5:1   542        1025: STR := 'WORKFILE IS SAVED';
557   1    5:1   568        1026: STR := 'NO DIRECTORY ON VOL';
558   1    5:1   596        1027: STR := 'NO WORKFILE TO SAVE';
559   1    5:1   624        1028: STR := 'WILDCARD NOT ALLOWED';
560   1    5:1   653        1029: STR := 'BAD FORMAT (WILDCARD <TO> NON-WILDCARD)';
561   1    5:1   701        1030: STR := 'ILLEGAL FILE/VOL NAME';
562   1    5:1   731        1031: STR := 'ILLEGAL CHANGE (VOL <TO> FILE) NAME';
563   1    5:1   775        1032: STR := 'BAD DEST FOR FILES FOUND'
564   1    5:1   777      END;
565   1    5:1   860      IF NUMBER IN [1017..1023] THEN
566   1    5:2   871        STR := CONCAT(STR,' EXPECTED');
567   1    5:1   906      CLEARLINE;
568   1    5:1   909      IF (NUMBER > 10) AND (NUMBER < 1000) OR SYSCOM^.MISCINFO.SLOWTERM THEN
569   1    5:2   929        WRITE('I/O ERROR #',NUMBER)  [ MISC. I/O ERROR. PRINT OUT ERROR # ONLY ]
570   1    5:1   958      ELSE
571   1    5:2   960        WRITE(STR);
```

```
572   1     5:1   969      IF EXXIT THEN
573   1     5:2   972        EXIT(CALLPROC)
574   1     5:0   976    END;
575   1     5:0   996
576   1     5:0   996    { CHECKS FOR SELECTED I/O ERRORS. WILL PRINT OUT ERROR AND }
577   1     5:0   996    { RETURN TO FILER PROMPT LINE IF ONE IS FOUND              }
578   1     6:0     1    PROCEDURE CHECKRSLT(RSLT : INTEGER);
579   1     6:0     0    BEGIN
580   1     6:1     0      IF (RSLT > 0) AND NOT (RSLT IN [13,14]) THEN
581   1     6:2    13        MESSAGES(RSLT,TRUE)
582   1     6:0    15    END;
583   1     6:0    30
584   1     6:0    30 {------------------------ WIDELY USED COMMAND SEQUENCES ----------------------}
585   1     6:0    30
586   1     6:0    30    { PERFORMS A WRITELN FOLLOWED BY A CLEARLINE }
587   1     7:0     1    PROCEDURE WRITEANDCLEAR;
588   1     7:0     0    BEGIN
589   1     7:1     0      WRITELN;
590   1     7:1     6      CLEARLINE
591   1     7:0     6    END;
592   1     7:0    22
593   1     7:0    22    { READS A CHARACTER FROM INPUT. RETURNS TRUE IF THE CHARACTER WAS A ('Y','Y')}
594   1     7:0    22    { FALSE OTHERWISE. EXITS TO PROMPT LINE IF THE CHARACTER WAS AN <ESC>. WILL  }
595   1     7:0    22    { POSITION CURSOR AT START OF NEXT LINE IF ALL WENT O.K.                     }
596   1     8:0     3    FUNCTION NGETCHAR(FLUSH : BOOLEAN) : BOOLEAN;
597   1     8:0     0    BEGIN
598   1     8:1     0      CH := GETCHAR(FLUSH);
599   1     8:1     8      IF (CH = SYSCOM^.CRTINFO.ALTMODE) THEN
600   1     8:2    20        EXIT(CALLPROC);
601   1     8:1    24      NGETCHAR := CH = 'Y';
602   1     8:1    29      IF NOT EOLN THEN
603   1     8:2    40        WRITELN
604   1     8:0    40    END;
605   1     8:0    58
606   1     8:0    58    { ASKS THE USER TO TYPE A SPACE TO CONTINUE. WILL RETURN TO THE FILER PROMPT }
607   1     8:0    58    { LINE IF THE USER RESPONDS WITH AN <ESC>. IF FLUSH THEN PRECLUDES TYPE-AHEAD}
608   1     9:0     1    PROCEDURE NSPACEWAIT(FLUSH : BOOLEAN);
609   1     9:0     0    BEGIN
610   1     9:1     0      IF SPACEWAIT(FLUSH) THEN
611   1     9:2     8        EXIT(CALLPROC)
612   1     9:0    12    END;
```

```
613   1     9:0     24
614   1     9:0     24   { USED TO UPDATE DIRECTORY AND CHECKS THE I/O RESULT }
615   1    10:0      1   PROCEDURE UPDATEDIR;
616   1    10:0      0   BEGIN
617   1    10:1      0     WRITEDIR(SOURCEUNIT,GDIR);
618   1    10:1      5     CHECKRSLT(IORESULT)
619   1    10:0      7   END;
620   1    10:0     22
621   1    10:0     22 {------------------ MISCELLANEOUS GRUNDGE PROCEDURES -------------------------}
622   1    10:0     22
623   1    10:0     22   { REMOVES SPACES AND UNPRINTABLE CHARACTERS FROM INPUT STRING. }
624   1    10:0     22   { CHANGES ALL LOWER-CASE CHARACTERS TO UPPER-CASE              }
625   1    11:0      1   PROCEDURE EATSPACES(VAR STRG : LONGSTRING);
626   1    11:0      2   VAR
627   1    11:0      2     I : INTEGER;
628   1    11:0      3
629   1    11:0      0   BEGIN
630   1    11:1      0     I := 1;
631   1    11:1      3     WHILE I <= LENGTH(STRG) DO
632   1    11:2     10       IF (ORD(STRG[I]) >= 33) AND (ORD(STRG[1]) <= 125) THEN
633   1    11:3     23         BEGIN
634   1    11:4     23           IF (STRG[I] >= 'A') AND (STRG[I] <= 'Z') THEN
635   1    11:5     36             STRG[I] := CHR( ORD( STRG[I] ) - ORD( 'A' ) + ORD ('A' ));
636   1    11:4     46           I := I + 1
637   1    11:3     47         END
638   1    11:2     51       ELSE
639   1    11:3     53         DELETE(STRG,I,1);
640   1    11:1     61     IF STRG = '' THEN
641   1    11:2     69       EXIT(CALLPROC);
642   1    11:0     73   END;
643   1    11:0     88
644   1    11:0     88   { ASCERTAINS THE CORRECT BLOCK NUMBER FOR PROCEDURES TO USE AT A GIVEN TIME }
645   1    11:0     88   { IF A VALID DEOVBLK EXISTS ON THE PRESENT DIRECTORY THEN THE USER WILL BE  }
646   1    11:0     88   { ASKED IF THAT VALUE IS THE CORRECT ONE. OTHERWISE A VALID BLOCK MUST BE   }
647   1    11:0     88   { ENTERED. FOR A BLOCK TO BE VALID IT MUST BE >= LASTBLK                    }
648   1    12:0      1   PROCEDURE GETBLOCKS(MESS1,MESS2,MESS3 : SHORTSTRING; LASTBLK: INTEGER;
649   1    12:0      5                                           VAR NBLOCKS: INTEGER);
650   1    12:0     45   VAR
651   1    12:0     45     OK : BOOLEAN;
652   1    12:0      0   BEGIN
653   1    12:1      0     OK := FALSE;
```

```
654   1   12:1    18      IF GDIR <> NIL THEN
655   1   12:2    23        IF GDIR^[0].DEOVBLK >= LASTBLK THEN
656   1   12:3    32          BEGIN
657   1   12:4    32            CLEARLINE;
658   1   12:4    35            WRITE(MESS1,' ',GDIR^[0].DEOVBLK,' ',MESS2,' ? (Y/N) ');
659   1   12:4   100            OK := NGETCHAR(TRUE);
660   1   12:4   107            NBLOCKS := GDIR^[0].DEOVBLK
661   1   12:3   112          END;
662   1   12:1   114      IF NOT OK THEN
663   1   12:2   119        BEGIN
664   1   12:3   119          CLEARLINE;
665   1   12:3   122          WRITE(MESS3,' ? ');
666   1   12:3   144          READLN(NBLOCKS);
667   1   12:3   157          IF NBLOCKS < LASTBLK THEN
668   1   12:4   163            BEGIN
669   1   12:5   163              CLEARLINE;
670   1   12:5   166              WRITE('INVALID #');
671   1   12:5   185              EXIT(CALLPROC)
672   1   12:4   189            END;
673   1   12:2   189        END
674   1   12:0   189    END;
675   1   12:0   202
676   1   12:0   202    [ ASCERTAINS IF THE USER REALLY WANTS TO DESTROY THE DIRECTORY OF A DISK  ]
677   1   12:0   202    [ IF THE USER DOESN'T THIS PROCEDURE WILL RETURN TO THE FILER PROMPT LINE ]
678   1   13:D     1    PROCEDURE RISKVOLUME;
679   1   13:0     0    BEGIN
680   1   13:1     0      IF (LASTSTATE = OKDIR) AND (GDIR <> NIL) THEN
681   1   13:2     9        BEGIN
682   1   13:3     9          CLEARLINE;
683   1   13:3    12          WRITE('DESTROY ',GVID,': ? ');
684   1   13:3    53          IF NOT NGETCHAR(TRUE) THEN
685   1   13:4    61            EXIT(CALLPROC)
686   1   13:2    65        END
687   1   13:0    65    END;
688   1   13:0    78
689   1   13:0    78
690   1   13:0    78 [$I FILER.A.TEXT]
690   1   13:0    78 [$I FILER.B.TEXT]
691   1   13:0    78 [      COPYRIGHT (C) 1979 REGENTS OF THE UNIVERSITY OF CALIFORNIA.      ]
692   1   13:0    78 [      PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-       ]
693   1   13:0    78 [      TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE      ]
```

```
694   1   13:0    73 [     OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.          ]
695   1   13:0    73
696   1   13:0    78 [---------------------- SPECIALIZED FILER ROUTINES -------------------------]
697   1   13:0    78
698   1   13:0    78   [ ASCERTAINS WHETHER OR NOT THE PROPER DISK IS IN THE PROPER DRIVE ]
699   1   13:0    78   [ IF IT IS NOT WILL ASK USER TO PUT THE DISK IN THE PROPER DRIVE   ]
700   1   13:0    78   [ IF THE USER DOES NOT DO SO THIS PROCEDURE WILL RETURN TO THE     ]
701   1   13:0    78   [ FILER PROMPT LINE                                               ]
702   1   14:0     1   PROCEDURE INSERTVOLUME(INTUNIT : INTEGER; VID1 : VID; CHECK : BOOLEAN);
703   1   14:0     8   VAR
704   1   14:0     8     OK : BOOLEAN;
705   1   14:0     9     OLDUNIT, NEWUNIT : VID;
706   1   14:0    17
707   1   14:0     0   BEGIN [ INSERTVOLUME ]
708   1   14:1     0     OLDUNIT := '#  ';
709   1   14:1    15     IF (INTUNIT DIV 10) = 1 THEN
710   1   14:2    22       OLDUNIT[2] := '1';
711   1   14:1    27     OLDUNIT [3] := CHR(ORD('0') + INTUNIT MOD 10);
712   1   14:1    36     EATSPACES(OLDUNIT);
713   1   14:1    40     OK := CHECK;
714   1   14:1    43     IF CHECK THEN   [ NEED TO MAKE SURE THE DISK IS IN THE DRIVE ]
715   1   14:2    46       IF VOLSEARCH(VID1,TRUE,GDIR) <> INTUNIT THEN  [ VOLUME IN PROPER DRIVE ]
716   1   14:2    60
717   1   14:2    60         [ KLUDGE !!!!!! FORCE THE OP-SYSTEM TO LOOK AT THE CORRECT UNIT ]
718   1   14:2    60         [ IF THERE ARE TWO VOLS WITH THE SAME NAME ON LINE IT WON'T BE  ]
719   1   14:2    60         [ ABLE TO FIND THE ONE ON THE LOWER DRIVE OTHERWISE             ]
720   1   14:2    60
721   1   14:3    60         BEGIN   [ VOLUME WAS NOT IN PROPER DRIVE, WHERE IS IT ? ]
722   1   14:4    60           NEWUNIT := OLDUNIT;
723   1   14:4    66           OK := VOLSEARCH(NEWUNIT,TRUE,GDIR) <> 0; [ 0 MEANS UNIT NOT FOUND ]
724   1   14:4    80           OK := OK AND (NEWUNIT = VID1)  [ IS THIS THE CORRECT VOLUME ? ]
725   1   14:3    87         END;
726   1   14:1    90       IF NOT OK THEN
727   1   14:1    94
728   1   14:1    94         [ REPEAT THE ABOVE AFTER ASKING THE USER TO PUT IN THE CORRECT DISK ]
729   1   14:1    94
730   1   14:2    94         BEGIN
731   1   14:3    94           CLEARLINE;
732   1   14:3    97           WRITELN('PUT ',VID1,': IN UNIT ',OLDUNIT);
733   1   14:3   155           NSPACEWAIT(TRUE);
734   1   14:3   158           IF CHECK THEN
```

```
735   1   14:4   161            BEGIN
736   1   14:5   161              OK := (VOLSEARCH(OLDUNIT,TRUE,GDIR) <> 0);
737   1   14:5   175              IF (NOT OK) OR (OLDUNIT <> VID1) THEN
738   1   14:6   186                EXIT(CALLPROC)
739   1   14:4   190            END
740   1   14:2   190        END
741   1   14:0   190    END [ INSERTVOLUME ];
742   1   14:0   202
743   1   14:0   202    [ SCANS THROUGH DIRMAP FOR FILES TO BE DELETED AND UPDATES THE  ]
744   1   14:0   202    [ DIRECTORY ON THE SOURCE UNIT CORRESPONDINGLY                  ]
745   1   15:0     1    PROCEDURE ZAPENTRIES(DIRMAP : BITMAP; UPDATE : BOOLEAN);
746   1   15:0     9    VAR
747   1   15:0     9      LOC : INTEGER;
748   1   15:0     0    BEGIN
749   1   15:1     0      IF DIRMAP.ENTRIES > 0 THEN
750   1   15:2    14        BEGIN
751   1   15:2    14
752   1   15:2    14          [ MAKE SURE THAT THE CORRECT DISK IS IN THE DRIVE ]
753   1   15:3    14          INSERTVOLUME(SOURCEUNIT,SOURCEVID,TRUE);
754   1   15:3    20
755   1   15:3    20          IF GDIR <> NIL THEN
756   1   15:4    25            BEGIN
757   1   15:5    25              FOR LOC := GDIR^ [0].DNUMFILES DOWNTO 1 DO
758   1   15:6    41                IF DIRMAP.DIRENTRY [LOC] THEN
759   1   15:7    50                  DELENTRY(LOC,GDIR);  [ DELETES FILE AT LOC IN THE DIRECTORY ]
760   1   15:5    62              IF UPDATE THEN
761   1   15:6    65                UPDATEDIR                    [ WRITES THE DIRECTORY OUT TO DISK     ]
762   1   15:4    65            END
763   1   15:2    67        END;
764   1   15:0    67    END;
765   1   15:0    82
766   1   15:0    82    [ PURGES THE FILE REQUESTED BY NAME FROM THE DIRECTORY. IF THE ]
767   1   15:0    82    [ FILE EXISTS AND MESS <> '' THEN WILL ASK YOU TO CONFIRM      ]
768   1   16:0     3    FUNCTION PURGEIT(NAME,MESS : SHORTSTRING):BOOLEAN;
769   1   16:0    31    VAR
770   1   16:0    31      GFIB : FILE;
771   1   16:0     0    BEGIN
772   1   16:1     0      RESET(GFIB,NAME);
773   1   16:1    31      PURGEIT := IORESULT = 0;
774   1   16:1    37      IF IORESULT = 0 THEN  [ RESULT OF 0 MEANS THAT THE FILE WAS FOUND ]
775   1   16:2    43        BEGIN
```

```
776   1   16:3    43          IF MESS <> '' THEN
777   1   16:4    52            BEGIN
778   1   16:5    52              CLEARLINE;
779   1   16:5    55              WRITE(MESS,' ',NAME,' ? ');
780   1   16:5    94              IF NOT NGETCHAR(TRUE) THEN
781   1   16:6   102                BEGIN
782   1   16:7   102                  PURGEIT := FALSE;       [ USER DOES NOT WISH TO REMOVE THE FILE ]
783   1   16:7   105                  EXIT(PURGEIT)
784   1   16:6   109                END
785   1   16:4   109            END;
786   1   16:3   109          CLOSE(GFIB,PURGE);
787   1   16:3   115          CHECKRSLT(IORESULT)
788   1   16:2   117        END;
789   1   16:0   119    END;
790   1   16:0   138
791   1   16:0   138    [ LETS USER KNOW WHAT IS BEING DONE TO HIS FILE ]
792   1   17:0     1    PROCEDURE PRINTMESS(VID1 : VID; TID1 : TID; DEST : SHORTSTRING);
793   1   17:0     0    BEGIN
794   1   17:1     0      CLEARLINE;
795   1   17:1    18      WRITE(VID1,':',TID1);
796   1   17:1    44      IF (LENGTH(DEST) + 31) > SYSCOM^.CRTINFO.WIDTH THEN
797   1   17:2    58        WRITEANDCLEAR
798   1   17:1    58      ELSE
799   1   17:2    62        IF NOT SYSCOM^.MISCINFO.SLOWTERM THEN
800   1   17:3    73          WRITE('':24-(LENGTH(VID1)+LENGTH(TID1)));
801   1   17:1    93      WRITELN(' --> ',DEST)
802   1   17:0   123    END;
803   1   17:0   136
804   1   17:0   136 [----------------- COMMAND PARSERS & DIRECTORY SEARCH ROUTINES ----------------]
805   1   17:0   136
806   1   17:0   136    [ ASCERTAINS THE EXACT STATE OF GTITLE. IF THAT STATE DOES NOT CORRESPOND ]
807   1   17:0   136    [ WITH THE ACCEPTABLE STATES FOR THAT STRING THEN AN APPROPRIATE ERROR    ]
808   1   17:0   136    [ MESSAGES WILL BE RETURNED TO THE USER AND RETURN TO THE FILER PROMPT    ]
809   1   17:0   136    [ LINE. OTHERWISE, WILL SET LASTSTATE TO THE CONDITION FOUND              ]
810   1   18:0     1    PROCEDURE SCANINPUT(GTITLE : STRNG; CHECK : CHCKS;
811   1   18:0     3                                ERROR : INTEGER; WHERE : LOCATION; GETDIR : BOOLEAN);
812   1   18:0    27    VAR
813   1   18:0    27      NEWDIR : ^INTEGER;
814   1   18:0    28      GSEGS : INTEGER;
815   1   18:0    29
816   1   19:0     1      PROCEDURE MAKECALL(ERR : INTEGER; STATE : CHECKS);
```

```
817   1   19:0     3      VAR
818   1   19:0     3        PRINTERROR : BOOLEAN;
819   1   19:0     4
820   1   19:0     0      BEGIN
821   1   19:1     0        LASTSTATE := STATE;
822   1   19:1     3        IF (STATE IN CHECK) OR (ERROR = 0) THEN
823   1   19:2    17          EXIT(SCANINPUT);
824   1   19:1    21        IF GTITLE = '' THEN
825   1   19:2    31          EXIT(CALLPROC);
826   1   19:1    35        HOMECURSOR;
827   1   19:1    38        WRITEANDCLEAR;
828   1   19:1    40
829   1   19:1    40        [ WRITES OUT THE EXPLICIT VOLUME NAME ]
830   1   19:1    40        IF GTITLE [1] = '*' THEN
831   1   19:2    49          BEGIN
832   1   19:3    49            DELETE(GTITLE,1,1);
833   1   19:3    57            IF GTITLE [1] = ':' THEN
834   1   19:4    66              DELETE(GTITLE,1,1);
835   1   19:3    74            WRITE(SYVID,':')
836   1   19:2    92          END
837   1   19:1    92        ELSE
838   1   19:2    94          IF GTITLE [1] = ':' THEN
839   1   19:3   103            WRITE(DKVID);
840   1   19:3   113
841   1   19:1   113        WRITE(GTITLE);
842   1   19:1   123        IF SYSCOM^.CRTINFO.WIDTH >= 80 THEN
843   1   19:2   132          WRITE(' - ')
844   1   19:1   145        ELSE
845   1   19:2   147          WRITEANDCLEAR; [ WRITE STRING IN ERROR ]
846   1   19:1   149        MESSAGES(ERR,FALSE);                [ WRITE THE STATE OF THE STRING ]
847   1   19:1   153
848   1   19:1   153        [ WAS THE USER EVEN CLOSE TO THE CORRECT FORMAT ]
849   1   19:1   153        CASE STATE OF
850   1   19:1   156          BADTITLE : PRINTERROR := TRUE;
851   1   19:1   161          NOVOL,BADUNIT,BADDIR : PRINTERROR := (ERROR=FILEEXP) AND (GTID='');
852   1   19:1   180          UNBLKDVOL : PRINTERROR := ERROR IN [BLKDEXP,FILEEXP,FILEBLKDEXP];
853   1   19:1   202          OKFILE,BADFILE : PRINTERROR := ERROR IN [BLKDEXP,UNBLKDEXP,VOLEXP];
854   1   19:1   224          OKDIR : PRINTERROR := ERROR IN [UNBLKDEXP,FILEEXP,FILEUNBLKDEXP]
855   1   19:1   241        END;
856   1   19:1   270        IF NOT SYSCOM^.MISCINFO.SLOWTERM THEN
857   1   19:2   281          BEGIN
```

```
858   1   19:3   231            IF PRINTERROR THEN
859   1   19:4   284              BEGIN
860   1   19:5   284                IF SYSCOM^.CRTINFO.WIDTH < 80 THEN
861   1   19:6   293                  WRITEANDCLEAR
862   1   19:5   293                ELSE
863   1   19:6   297                  WRITE(', ');
864   1   19:5   309                MESSAGES(ERROR,FALSE)
865   1   19:4   313              END;
866   1   19:4   315
867   1   19:4   315            [ USER MAY NEED TO KNOW WHICH PART OF THE STRING IS IN ERROR ]
868   1   19:3   315            IF WHERE = SOURCE THEN
869   1   19:4   322              WRITE(' <SOURCE>');
870   1   19:3   341            IF WHERE = DESTINATION THEN
871   1   19:4   348              WRITE(' <DEST>');
872   1   19:2   365          END;
873   1   19:1   365        EXIT(CALLPROC)
874   1   19:0   369      END;
875   1   19:0   382
876   1   18:0     0    BEGIN [ SCANINPUT ]
877   1   18:1     0      GUNIT := 0;
878   1   18:1     8      IF SCANTITLE(GTITLE,GVID,GTID,GSEGS,GKIND) THEN [ BREAK UP INPUT STRING ]
879   1   18:2    25        BEGIN
880   1   18:3    25          IF GETDIR THEN
881   1   18:4    28            MARK(NEWDIR);     [ WILL CAUSE THE PRESENT DIRECTORY TO DISSAPPEAR ]
882   1   18:3    32          GVID2 := GVID;      [ SAVE PRESENT GVID ]
883   1   18:3    38          GUNIT := VOLSEARCH(GVID,TRUE,GDIR);  [ SEARCHS FOR PROPER VOLUME ]
884   1   18:3    50          IF GDIR = NIL THEN   [ WASN'T ABLE TO READ A DIRECTORY OFF THE VOLUME ]
885   1   18:4    55            BEGIN
886   1   18:5    55              IF GUNIT = 0 THEN
887   1   18:6    60                MAKECALL(9,NOVOL);   [ NO SUCH VOL WAS ON-LINE ]
888   1   18:5    64              UNITCLEAR(GUNIT);
889   1   18:5    67              IF IORESULT <> 0 THEN
890   1   18:6    73                MAKECALL(2,BADUNIT); [ BAD UNIT # GIVEN ]
891   1   18:5    77              IF UNITABLE [GUNIT].UISBLKD THEN
892   1   18:6    86                MAKECALL(NODIR,BADDIR);  [ VOL WAS BLKD, BUT NO DIR WAS ON IT ]
893   1   18:5    92              MAKECALL(UNBLKD,UNBLKDVOL) [ VOLUME WAS NOT BLOCKED ]
894   1   18:4    96            END;
895   1   18:3    98          IF GTID = '' THEN
896   1   18:4   107            MAKECALL(BLKD,OKDIR);  [ VOL WAS BLKD & THE DIR IS OK ]
897   1   18:3   113          IF DIRSEARCH(GTID,TRUE,GDIR) <> 0 THEN
898   1   18:4   126            MAKECALL(FOUNDFILE,OKFILE);   [ THE FILE WAS FOUND ]
```

```
899   1   18:3   132          MAKECALL(10,BADFILE) [ THE FILE WAS NOT FOUND ]
900   1   18:2   134        END;
901   1   18:1   136      MAKECALL(ILLFILEVOL,BADTITLE) [ ILLEGAL NAME (TOO LONG, OR MISSING BRACKET ]
902   1   18:0   140  END [ SCANINPUT ];
903   1   18:0   154
904   1   18:0   154  [ DIRECTORY SEARCH ROUTINE FOR FINDING THE USER REQUESTED FILES ]
905   1   18:0   154  [ ON THE FIRST CALL TO THIS ROUTINE ALL TABLES AND NECCESSARY   ]
906   1   18:0   154  [ BITS IN THE DIRECTORY WILL BE UPDATED TO KEEP TRACK OF THE    ]
907   1   18:0   154  [ NESCESSARY FILES, WITHOUT LOSING ANY. ALL FILES THAT ARE TO   ]
908   1   18:0   154  [ BE USED IN WILDCARD OPERATIONS MUST BE PRESENT ON THE INITIAL ]
909   1   18:0   154  [ INITIAL CALL TO THIS ROUTINE                                 ]
910   1   20:0     3  FUNCTION SEARCHDIR(MESSAGE : STRNG; VAR GINX : INTEGER;
911   1   20:0     5                                DEST, SCREENCLEAR : BOOLEAN) : BOOLEAN;
912   1   20:0    28  VAR
913   1   20:0    28    X : INTEGER;
914   1   20:0    29    NEWSTRING : TID;
915   1   20:0    37
916   1   20:0    37      [ AT THIS POINT A REQUESTED FILE HAS BEEN FOUND. IN CASE THAT    ]
917   1   20:0    37      [ QUESTION IS TRUE WE MUST SEE IF THE USER STILL WANTS TO USE IT ]
918   1   21:0     1      PROCEDURE FOUNDFILE;
919   1   21:0     0      BEGIN
920   1   21:1     0        WITH GDIR^ [GINX] DO
921   1   21:2     9          BEGIN
922   1   21:3     9            SOURCETITLE := DTID;
923   1   21:3    16            FROMWHERE := CONCAT(VOLNAME1,':',DTID);
924   1   21:3    53            CH := 'Y';
925   1   21:3    56            FOUND := FILEFOUND;     [ YES A USABLE FILE HAS BEEN FOUND ]
926   1   21:3    59            IF (MESSAGE <> '') AND QUESTION THEN [ CONFIRM OPERATION ]
927   1   21:4    71              BEGIN
928   1   21:5    71                CLEARLINE;
929   1   21:5    74                IF NOT SYSCOM^.MISCINFO.SLOWTERM THEN
930   1   21:6    85                  WRITE(MESSAGE,' ');
931   1   21:5   103                WRITE(DTID,' ? ');
932   1   21:5   126                CH := GETCHAR(FALSE);
933   1   21:5   134                IF NOT EOLN THEN
934   1   21:6   145                  WRITELN;
935   1   21:5   151                IF CH = SYSCOM^.CRTINFO.ALTMODE THEN [ USER WANTS TO ABORT ]
936   1   21:5   163
937   1   21:5   163                  [ DON'T RETURN TO PROMPT LINE BECAUSE OF THE R(EMOVE COMMAND ]
938   1   21:6   163                  BEGIN
939   1   21:7   163                    FOUND := ABORTIT;
```

```
940   1   21:7   166                    EXIT(SEARCHDIR)
941   1   21:6   170                  END
942   1   21:4   170                END
943   1   21:2   170              END;
944   1   21:1   170            SEARCHDIR := CH = 'Y'
945   1   21:0   171          END;
946   1   21:0   168
947   1   21:0   188    [ CHECKS TO SEE IF THE REQUESTED PORTION OF THE TWO STRINGS MATCH ]
948   1   22:0     3    FUNCTION TESTSTR(STR : TID; START : INTEGER) : BOOLEAN;
949   1   22:0    13    VAR
950   1   22:0    13      TEMP : TID;
951   1   22:0     0    BEGIN
952   1   22:1     0      TEMP [0] := STR [0];
953   1   22:1    13      MOVELEFT(GDIR^ [X].DTID[START],TEMP[1],LENGTH(STR));
954   1   22:1    31      TESTSTR := TEMP = STR
955   1   22:0    33    END;
956   1   22:0    52
957   1   20:0     0    BEGIN [ SEARCHDIR ]
958   1   20:1     0      SEARCHDIR := FALSE;
959   1   20:1     8      IF GINX = 0 THEN
960   1   20:2    14        BEGIN
961   1   20:3    14          DEST := DEST AND UNITABLE [DESTUNIT].UISBLKD;
962   1   20:3    25          IF SCREENCLEAR AND WILDCARD THEN
963   1   20:4    30            BEGIN
964   1   20:5    30              CLEARSCREEN;
965   1   20:5    33              WRITELN
966   1   20:4    33            END;
967   1   20:3    39          FOUND := NOFILES;
968   1   20:3    42
969   1   20:3    42          [ WILL IT BE NESCESSARY TO USE THE STATUS BITS IN THE DIRECTORY ]
970   1   20:3    42          [ TO KEEP PROPER TRACK OF THE FILES                             ]
971   1   20:3    42          MARKING := DEST AND (SOURCEVID = DESTVID);
972   1   20:3    52
973   1   20:3    52          [ SEARCH DIRECTORY FOR ELIGIBLE SOURCE FILES ]
974   1   20:3    52          FOR X := 1 TO GDIR^ [0].DNUMFILES DO
975   1   20:4    70          WITH GDIR^ [X] DO
976   1   20:5    77            BEGIN
977   1   20:6    77              STATUS := FALSE;
978   1   20:6    85              IF (LENGTH(STRING1) + LENGTH(STRING2)) <= LENGTH(DTID) THEN
979   1   20:7   104                IF TESTSTR(STRING1,1) AND
980   1   20:7   112                  TESTSTR(STRING2,LENGTH(DTID) - LENGTH(STRING2) + 1) THEN
```

```
 981   1   20:8   134                WITH DIRMAP DO
 982   1   20:9   134                BEGIN  [ THIS FILE MATCHES THE NESCESSARY STRINGS ]
 983   1   20:0   134                  IF (STRING1=DTID) OR WILDCARD THEN
 984   1   20:1   147                    BEGIN
 985   1   20:2   147                      STATUS := MARKING;
 986   1   20:2   155                      DIRENTRY [X] := TRUE;
 987   1   20:2   164                      ENTRIES := ENTRIES + 1;
 988   1   20:1   176                    END;
 989   1   20:0   176                  FOUND := FILESNOGOOD
 990   1   20:9   176                END;
 991   1   20:5   179          END;
 992   1   20:3   187        IF MARKING THEN
 993   1   20:4   190          UPDATEDIR      [ MUST MAINTAIN THE STATUS BITS IN THE DIRECTORY ]
 994   1   20:2   190      END
 995   1   20:1   192    ELSE
 996   1   20:2   194      IF DEST THEN
 997   1   20:3   197        GINX := GINX - 1;
 998   1   20:1   203    IF DIRMAP.ENTRIES > 0 THEN
 999   1   20:2   212    BEGIN
1000   1   20:3   212      INSERTVOLUME(SOURCEUNIT,SOURCEVID,TRUE); [ GET THE SOURCE VOLUME ON-LINE ]
1001   1   20:3   218      CH := 'N';
1002   1   20:3   221      WHILE (GINX < GDIR^[0].DNUMFILES) AND (CH <> 'Y') DO
1003   1   20:4   236        WITH GDIR^ [GINX+1] DO
1004   1   20:5   245          BEGIN
1005   1   20:6   245            GINX := GINX + 1; [ LOOK AT THE NEXT DIRECTORY ENTRY ]
1006   1   20:6   251            IF MARKING THEN
1007   1   20:7   254              BEGIN
1008   1   20:8   254                DIRMAP.DIRENTRY [GINX] := STATUS;
1009   1   20:8   269                IF STATUS THEN
1010   1   20:9   278                  BEGIN
1011   1   20:0   278                    STATUS := FALSE;   [ TURN OFF STATUS BIT IN DIRECTORY ]
1012   1   20:0   286                    UPDATEDIR
1013   1   20:9   286                  END
1014   1   20:7   288              END;
1015   1   20:6   288            IF DIRMAP.DIRENTRY [GINX] THEN
1016   1   20:7   298              BEGIN    [ SOURCE FILE FOR THIS ENTRY IS O.K. WHAT ABOUT DEST ]
1017   1   20:8   298                IF DEST THEN
1018   1   20:9   301                  BEGIN
1019   1   20:0   301                    NEWSTRING := COPY(DTID,LENGTH(STRING1) + 1,
1020   1   20:0   316                            LENGTH(DTID) - LENGTH(STRING1) - LENGTH(STRING2));
1021   1   20:0   340                    X := LENGTH(NEWSTRING) +
```

```
1022   1   20:0   344                    SCAN(LENGTH(STRING3),= '[',STRING3[1]) + LENGTH(STRING4);
1023   1   20:0   364                  IF (X <= TIDLENG) AND ((X > 0) OR NOT WILDCARD) THEN
1024   1   20:0   378
1025   1   20:0   378                    [ DESTINATION FILE WILL BE O.K. (ITS SMALL ENOUGH ) ]
1026   1   20:1   378                    BEGIN
1027   1   20:2   378                      TOWHERE := CONCAT(VOLNAME2,':',STRING3,
1028   1   20:2   412                                                NEWSTRING,STRING4);
1029   1   20:2   430                      IF (STRING1 = DTID) OR WILDCARD THEN
1030   1   20:3   443                        FOUNDFILE
1031   1   20:1   443                    END
1032   1   20:0   445                  ELSE
1033   1   20:1   447                    PRINTMESS(SOURCEVID,DTID,'NOT PROCCESSED');
1034   1   20:9   472                END
1035   1   20:8   472              ELSE
1036   1   20:9   474                FOUNDFILE;  [ NO DESTINATION FILE IS NEEDED ]
1037   1   20:8   476              DIRMAP.DIRENTRY [GINX] := FALSE; [ TURN OFF BIT FOR THIS ENTRY ]
1038   1   20:8   485              DIRMAP.ENTRIES := DIRMAP.ENTRIES - 1; [ ONE LESS ENTRY TO DO ]
1039   1   20:7   497            END;
1040   1   20:5   497        END;
1041   1   20:2   499    END;
1042   1   20:1   499    IF FOUND = NOFILES THEN
1043   1   20:2   505      MESSAGES(ORD(INOFILE),FALSE);  [ NO REQUESTED FILES WERE FOUND ]
1044   1   20:1   509    IF FOUND = FILESNOGOOD THEN
1045   1   20:2   515      MESSAGES(BADDEST,FALSE); [ THE REQ. FILES FOUND COULD NOT BE USED ]
1046   1   20:0   521  END [ SEARCHDIR ];
1047   1   20:0   548
1048   1   20:0   548  [ INPUT STRING PARSER. REMOVES WILDCARD SYMBOLS. SETS WILDCARD AND QUESTION ]
1049   1   20:0   548  [ EXPANDS DOLLAR SIGNS. SETS SOURCEVID, DESTVID, SOURCEUNIT, DESTUNIT,      ]
1050   1   20:0   548  [ VOLNAME1, VOLNAME2, STRING1, STRING2, STRING3, STRING4                    ]
1051   1   23:D     1  PROCEDURE CHECKFILE(MSG1,MSG2 : SHORTSTRING; DEFAULT,ERROR1 : INTEGER;
1052   1   23:D     5                       WILD,FILLE : BOOLEAN; CHECK1 : CHCKS);
1053   1   23:D    34  VAR
1054   1   23:D    34    SRCSTR : STRING;
1055   1   23:D    75
1056   1   23:D    75    [ WILL SCAN UP TO THE NEXT ',' OR TO THE END OF THE LINE. DOES ALL '$' ]
1057   1   23:D    75    [ EXPANSIONS. PARSES STRING FOR WILDCARDS, VOLNAME & FILENAME. MAKES   ]
1058   1   23:D    75    [ SURE THAT THE SOURCE AND DEST FILES ARE OF THE APPROPRIATE CLASS AND ]
1059   1   23:D    75    [ THAT NEEDED VOLUMES STAY ON LINE                                     ]
1060   1   24:D     1    PROCEDURE PROCESSDATA(MSG:SHORTSTRING; FIRSTCALL:BOOLEAN; VAR VOLNAME:VID;
1061   1   24:D     4                          VAR FIRSTSTR,SECONDSTR:SHORTSTRING; VAR WHERETO:STRNG);
1062   1   24:D    20    VAR
```

```
1063   1   24:0    20      STR,OLD : STRING;
1064   1   24:0   102      WHERE : LOCATION;
1065   1   24:0   103      X,I,LOC : INTEGER;
1066   1   24:0   106
1067   1   24:0   106      [ MAKES SURE THAT THE STRUCTURE BEFORE THE LAST DELIMMITER IS OF THE ]
1068   1   24:0   106      [ APPROPRIATE SIZE & CONTAINS NO SPECIAL SYMBOLS. IF FOR ANY REASON  ]
1069   1   24:0   106      [ THE STRING IS NOT CORRECT AN ERROR WILL BE FLAGGED AND THIS        ]
1070   1   24:0   106      [ PROCEDURE WILL RETURN TO THE FILER PROMPT LINE                     ]
1071   1   25:0     1      PROCEDURE FINDDELIM(SIZE,MESSAGE : INTEGER; VAR STRIING : STRNG);
1072   1   25:0     4      VAR
1073   1   25:0     4        ERROR : BOOLEAN;
1074   1   25:0     5
1075   1   25:0     5        [ SCANS STRIING FOR THE APPROPRIATE SPECIAL SYMBOL ['$','?','='] ]
1076   1   26:0     3        FUNCTION SCAN2(CH : CHAR) : BOOLEAN;
1077   1   26:0     0        BEGIN
1078   1   26:1     0          SCAN2 := SCAN(LOC,= CH,STRIING[1]) = LOC
1079   1   26:0    12        END;
1080   1   26:0    30
1081   1   25:0     0      BEGIN [ FINDDELIM ]
1082   1   25:1     0        STRIING := COPY(STR,1,LOC);    [ RETURNS CORRECT PORTION OF THE STRING ]
1083   1   25:1    17        ERROR := LOC > SIZE;           [ TOO LONG TO BE A LEGAL ENTRY ]
1084   1   25:1    24        DELETE(STR,1,LOC);
1085   1   25:1    34        DELETE(STR,1,1);
1086   1   25:1    42        IF (NOT ERROR) AND SCAN2('$') AND SCAN2('=') AND SCAN2('?') THEN
1087   1   25:2    64          EXIT(FINDDELIM);    [ NO ERRORS ENCOUNTERED ]
1088   1   25:1    68        CLEARLINE;
1089   1   25:1    71        IF NOT SYSCOM^.MISCINFO.SLOWTERM THEN
1090   1   25:2    82          WRITE(STRIING);
1091   1   25:1    90        IF SYSCOM^.CRTINFO.WIDTH < 80 THEN   [ LINE WILL NOT FIT IN 40 CHARS. ]
1092   1   25:2    99          WRITEANDCLEAR;
1093   1   25:1   101        IF ERROR THEN
1094   1   25:2   104          WRITE('...TOO LONG <');
1095   1   25:1   127        CASE MESSAGE OF
1096   1   25:1   130          1 : WRITE(' FILE NAME ');
1097   1   25:1   153          2 : WRITE(' SCAN STRING ');
1098   1   25:1   178          3 : WRITE(' VOL NAME ');
1099   1   25:1   200        END;
1100   1   25:1   214        IF ERROR THEN
1101   1   25:2   217          BEGIN
1102   1   25:3   217            IF MESSAGE = 3 THEN
1103   1   25:4   222              WRITE(VIDLENG)
```

```
1104   1   25:3   230            ELSE
1105   1   25:4   232              WRITE(TIDLENG);
1106   1   25:3   240            WRITE(' - CHAR. MAX >');
1107   1   25:2   264          END
1108   1   25:1   264        ELSE
1109   1   25:2   266          WRITE('- ILLEGAL FORMAT');
1110   1   25:1   292        EXIT(CALLPROC)
1111   1   25:0   296      END [ FINDDELIM ];
1112   1   25:0   308
1113   1   25:0   308      [ SCAN STR FOR SPECIAL SYMBOLS ['$','=','?',',',] ]
1114   1   27:0     3      FUNCTION SCAN1(CH : CHAR) : INTEGER;
1115   1   27:0     0      BEGIN
1116   1   27:1     0        SCAN1 := SCAN(LENGTH(STR), = CH,STR[1]);
1117   1   27:0    16      END;
1118   1   27:0    28
1119   1   24:0     0    BEGIN
1120   1   24:0     0      [ NEED TO GET INPUT STRING FROM USER ]
1121   1   24:1     0      IF INSTRING = '' THEN
1122   1   24:2    15        BEGIN
1123   1   24:3    15          CLEARLINE;
1124   1   24:3    18          WRITE(MSG);
1125   1   24:3    27          IF FIRSTCALL AND FAST THEN
1126   1   24:4    32            IF FILLE THEN
1127   1   24:5    37              WRITE(' WHAT FILE')
1128   1   24:4    57            ELSE
1129   1   24:5    59              WRITE(' WHAT VOL');
1130   1   24:3    78          WRITE(' ? ');
1131   1   24:3    91          READLN(INSTRING);
1132   1   24:3   109          EATSPACES(INSTRING)
1133   1   24:2   112        END;
1134   1   24:2   114
1135   1   24:2   114      [ COPY INPUT STRING INTO STR UP TO THE FIRST COMMA OR END OF LINE ]
1136   1   24:1   114      LOC := SCAN(LENGTH(INSTRING), = ',',INSTRING[1]);
1137   1   24:1   130      IF LOC > 35 THEN
1138   1   24:2   136        EXIT(CALLPROC);
1139   1   24:1   140      STR := COPY(INSTRING,1,LOC);
1140   1   24:1   157      DELETE(INSTRING,1,LOC);
1141   1   24:1   166      DELETE(INSTRING,1,1);
1142   1   24:1   174
1143   1   24:1   174      [ PARSE VOLUME NAME OUT OF STR. CHECK TO SEE IF QUESTION IS TRUE ]
1144   1   24:1   174      QUESTION := QUESTION OR (SCAN1('?') < LENGTH(STR));
```

```
1145   1   24:1   188      LOC := SCAN1(':');
1146   1   24:1   195      IF (STR[1] = '#') OR (LOC < LENGTH(STR)) THEN
1147   1   24:2   211        FINDDELIM(VIDLENG,3,VOLNAME)
1148   1   24:1   214      ELSE
1149   1   24:2   218        IF STR[1] = '*' THEN
1150   1   24:3   226          BEGIN
1151   1   24:4   226            DELETE(STR,1,1);
1152   1   24:4   233            VOLNAME := '*'
1153   1   24:3   234          END;
1154   1   24:1   237      WHERETO := CONCAT(VOLNAME,':');
1155   1   24:1   262
1156   1   24:1   262      [ EXPAND THE '$' IF ONE EXISTS     ]
1157   1   24:1   262      I := SCAN1('$');
1158   1   24:1   269      IF I < LENGTH(STR) THEN
1159   1   24:2   278        BEGIN
1160   1   24:3   278          IF LENGTH(STR)+LENGTH(SRCSTR)-1 > 35 THEN
1161   1   24:4   294            EXIT(CALLPROC);    [ ILLEGAL EXPANSION, TOO LONG ]
1162   1   24:3   298          OLD := STR;
1163   1   24:3   304          STR[0] := CHR(LENGTH(SRCSTR)+LENGTH(STR)-1);
1164   1   24:3   320          MOVELEFT(SRCSTR[1],STR[I+1],LENGTH(SRCSTR));
1165   1   24:3   337          MOVELEFT(OLD[I+2],STR[I+1+LENGTH(SRCSTR)],LENGTH(OLD)-I-1);
1166   1   24:2   366        END;
1167   1   24:2   366
1168   1   24:2   366      [ SCAN FOR WILDCARDS ]
1169   1   24:1   366      SRCSTR := STR;
1170   1   24:1   373      LOC := SCAN1('=');
1171   1   24:1   380      IF LOC = LENGTH(STR) THEN
1172   1   24:2   389        LOC := SCAN1('?');
1173   1   24:1   396      IF LOC < LENGTH(STR) THEN
1174   1   24:1   405
1175   1   24:1   405        [ WILCARD SYSMBOL IS PRESENT. PARSE REMAINING STRING ACCORDINGLY ]
1176   1   24:2   405        IF WILD THEN
1177   1   24:3   410          BEGIN
1178   1   24:4   410            IF NOT (FIRSTCALL OR WILDCARD) THEN
1179   1   24:5   416              MESSAGES(BADFORM,TRUE);
1180   1   24:4   422            WILDCARD := TRUE;
1181   1   24:4   425            FINDDELIM(TIDLENG,2,FIRSTSTR);
1182   1   24:4   430            LOC := LENGTH(STR);
1183   1   24:4   436            FINDDELIM(TIDLENG,2,SECONDSTR)
1184   1   24:3   439          END
1185   1   24:2   441        ELSE
```

```
1186   1   24:3   443          MESSAGES(NOWILD,TRUE)   [ WILCARD OPERATION IS NOT ALLOWED ]
1187   1   24:1   447       ELSE
1188   1   24:1   451
1189   1   24:1   451         [ NO WILCARDS. REMAINING STRING IS A STANDARD FILENAME ]
1190   1   24:2   451         BEGIN
1191   1   24:3   451           IF (NOT FIRSTCALL) AND WILDCARD AND (DEFAULT = 0) THEN
1192   1   24:3   463
1193   1   24:3   463             [ USER USED A WILDCARD SYMBOL FOR THE SOURCE FILE BUT NOT THE ]
1194   1   24:3   463             [ DESTINATION FILE. ONLY CASES THAT THIS IS ALLOWED IS WHEN   ]
1195   1   24:3   463             [ THE USER IS LISTING THE DIRECTORY (I.E., DEFAULT <> 0) OR   ]
1196   1   24:3   463             [ WHEN THE DESTINATION FILE IS AN UNBLKD-VOLUME               ]
1197   1   24:4   463             BEGIN
1198   1   24:5   463               SCANINPUT(CONCAT(VOLNAME2,':'),[],0,NEITHER,FALSE);
1199   1   24:5   494               IF LASTSTATE <> UNBLKDVOL THEN
1200   1   24:6   499                 MESSAGES(BADFORM,TRUE);
1201   1   24:4   505             END;
1202   1   24:3   505           FINDDELIM(SHSTRLENG,1,FIRSTSTR);
1203   1   24:3   510           WHERETO := CONCAT(WHERETO,FIRSTSTR)
1204   1   24:2   530         END;
1205   1   24:1   532       IF NOT FIRSTCALL THEN
1206   1   24:1   536
1207   1   24:1   536         [ SET DESTUNIT & DESTVID TO THERE PROPER VALUES ]
1208   1   24:2   536         BEGIN
1209   1   24:3   536           SCANINPUT(WHERETO,[],0,WHERE,TRUE);
1210   1   24:3   546           DESTUNIT := GUNIT;
1211   1   24:3   549           DESTVID := GVID;
1212   1   24:3   555
1213   1   24:3   555           [ MAKE SURE THAT THE USER HASN'T REMOVED THE SOURCE DISK ]
1214   1   24:3   555           INSERTVOLUME(SOURCEUNIT,SOURCEVID,TRUE);
1215   1   24:2   561         END
1216   1   24:1   561       ELSE
1217   1   24:2   563         IF (INSTRING <> '') AND (DEFAULT <= 0) THEN
1218   1   24:3   579           WHERE := DESTINATION
1219   1   24:2   579         ELSE
1220   1   24:3   584           WHERE := NEITHER;
1221   1   24:3   587
1222   1   24:3   587       [ RESTORE THE DIRECTORY ETC. FOR THE SOURCE VOLUME ]
1223   1   24:1   587       IF WILDCARD THEN
1224   1   24:2   590         SCANINPUT(FROMWHERE,[OKDIR],BLKDEXP,SOURCE,TRUE)
1225   1   24:1   602       ELSE
1226   1   24:2   606         SCANINPUT(FROMWHERE,CHECK1,ERROR1,SOURCE,TRUE);
```

```
1227   1    24:0    622      END;
1228   1    24:0    634
1229   1    23:0      0    BEGIN [ CHECKFILE ]
1230   1    23:1      0      INITGLOBALS;
1231   1    23:1     12      SRCSTR := '';
1232   1    23:1     19      PROCESSDATA(MSG1,TRUE,VOLNAME1,STRING1,STRING2,FROMWHERE);
1233   1    23:1     34      SOURCEUNIT := GUNIT;
1234   1    23:1     37      SOURCEVID := GVID;
1235   1    23:1     43      IF DEFAULT > 0 THEN
1236   1    23:2     48        EXIT(CHECKFILE);
1237   1    23:1     52      IF (INSTRING = '') AND (DEFAULT < 0) THEN
1238   1    23:2     66        EXIT(CHECKFILE);
1239   1    23:1     70      PROCESSDATA(MSG2,FALSE,VOLNAME2,STRING3,STRING4,TOWHERE);
1240   1    23:0     84    END;
1241   1    23:0     96
1242   1    23:0     96
1243   1    23:0     96
1244   1    23:0     96 [$I FILER.B.TEXT]
1244   1    23:0     96 [$I FILER.C.TEXT]
1245   1    23:0     96 [      COPYRIGHT (C) 1979 REGENTS OF THE UNIVERSITY OF CALIFORNIA.      ]
1246   1    23:0     96 [      PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-       ]
1247   1    23:0     96 [      TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE      ]
1248   1    23:0     96 [      OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.             ]
1249   1    23:0     96
1250   1    23:0     96 [------------ PROCEDURES FOR MOVING, MAKING AND CHANGING FILES ---------------]
1251   1    23:0     96
1252   1    23:0     96    [ CHECKS TO SEE IF FILE IS ENDANGERED BY THE OPERATION TO BE PERFORMED ]
1253   1    28:D      3    FUNCTION FINDSAME(DOO : BOOLEAN):BOOLEAN;
1254   1    28:0      0    BEGIN
1255   1    28:1      0      FINDSAME := TRUE;
1256   1    28:1      3      IF (LASTSTATE = OKFILE) AND (DOO OR (SOURCETITLE <> GTID) OR
1257   1    28:1     14                                              (SOURCEVID <> GVID)) THEN
1258   1    28:2     24        FINDSAME := PURGEIT(CONCAT(GVID,':',GTID),'REMOVE OLD')
1259   1    28:0     68    END;
1260   1    28:0     86
1261   1    28:0     86    [ ALLOWS THE USER TO CHANGE THE NAME OF ANY FILE IN THE DIRECTORY ]
1262   1    28:0     86    [ OR THE NAME OF ANY BLOCKED DEVICE                               ]
1263   1    29:D      1    PROCEDURE CHANGER;
1264   1    29:D      1    VAR
1265   1    29:D      1      GFIB : UNTYPED;
1266   1    29:0     41      GFIBP : FIBP;
```

```
1267   1   29:0    42     IORSLT,LOC : INTEGER;
1268   1   29:0    44
1269   1   29:0     0   BEGIN [ CHANGER ]
1270   1   29:1     0     REPEAT
1271   1   29:2    12       CHECKFILE('CHANGE','CHANGE TO WHAT',0,FILEBLKDEXP,TRUE,TRUE,
1272   1   29:2    44                                                    [OKFILE,OKDIR]);
1273   1   29:2    52       IF ((STRING1 <> '') AND (STRING3 <> '')) OR WILDCARD THEN
1274   1   29:2    72
1275   1   29:2    72         [ CHANGING A FILENAME ]
1276   1   29:3    72         BEGIN
1277   1   29:4    72           VOLNAME2 := VOLNAME1;   [ DEST VOLNAME MUST BE THE SAME AS SOURCE ]
1278   1   29:4    78           TOWHERE := CONCAT(VOLNAME1,':');
1279   1   29:4   106           IF NOT WILDCARD THEN
1280   1   29:5   110             TOWHERE := CONCAT(TOWHERE,STRING3);  [ DEST FILENAME IS IN STRING3 ]
1281   1   29:4   137           LOC := 0;
1282   1   29:4   140           WHILE SEARCHDIR('CHANGE',LOC,TRUE,TRUE) DO
1283   1   29:5   159             BEGIN
1284   1   29:6   159               RESET(GFIB,FROMWHERE);   [ OPENS FILE TO BE CHANGED ]
1285   1   29:6   169               CHECKRSLT(IORESULT);
1286   1   29:6   173               GFIBP := GETPTR(GFIB);   [ GETS THE POINTER TO THE FILES HEADER ]
1287   1   29:6   181               SCANINPUT(TOWHERE,[BADFILE,OKFILE],FILEEXP,DESTINATION,TRUE);
1288   1   29:6   197               IF FINDSAME(FALSE) THEN
1289   1   29:7   204                 WITH GFIBP^ DO
1290   1   29:8   208                   BEGIN
1291   1   29:9   208                     FHEADER.DACCESS.YEAR := 100; [ LET THE OP-SYSTEM KNOW ]
1292   1   29:9   216                     PRINTMESS(FVID,FHEADER.DTID,GTID);
1293   1   29:9   228                     FHEADER.DTID := GTID;        [ CHANGE THE FILENAME ]
1294   1   29:8   236                   END;
1295   1   29:6   236               CLOSE(GFIB);
1296   1   29:6   242               CHECKRSLT(IORESULT)
1297   1   29:5   244             END
1298   1   29:3   246         END
1299   1   29:2   248       ELSE
1300   1   29:3   250         IF LENGTH(STRING1) + LENGTH(STRING3) = 0 THEN
1301   1   29:3   264
1302   1   29:3   264           [ CHANGING A VOLUME NAME ]
1303   1   29:4   264           BEGIN
1304   1   29:5   264             SCANINPUT(TOWHERE,[NOVOL,OKDIR],BLKDEXP,DESTINATION,TRUE);
1305   1   29:5   278             IF LASTSTATE = OKDIR THEN
1306   1   29:6   283               MESSAGES(VOLONLINE,TRUE); [ DON'T ALLOW TWO VOLS WITH SAME NAME ]
1307   1   29:6   289
```

```
1308  1  29:6  299            [ ALLOCATE ROOM FOR THE DIRECTORY & READ IT IN ]
1309  1  29:5  289            NEW(GDIR);
1310  1  29:5  296            UNITREAD(SOURCEUNIT,GDIR^,SIZEOF(DIRECTORY),DIRBLK);
1311  1  29:5  306            CHECKRSLT(IORESULT);
1312  1  29:5  310            GDIR^[0].DVID := GVID;  [ CHANGE THE VOLUME NAME ]
1313  1  29:5  320            UNITWRITE(SOURCEUNIT,GDIR^,SIZEOF(DIRECTORY),DIRBLK);
1314  1  29:5  330            IORSLT := IORESULT;
1315  1  29:5  334            RELEASE(GDIR);
1316  1  29:5  338            CHECKRSLT(IORSLT);
1317  1  29:5  342            UNITABLE[SOURCEUNIT].UVID := GVID;   [ UPDATE THE UNITABLE ]
1318  1  29:5  352            IF (SYVID = SOURCEVID) AND (SYSCOM^.SYSUNIT = SOURCEUNIT) THEN
1319  1  29:6  368              SYVID := GVID;   [ NAME OF ROOT DEVICE HAS BEEN CHANGED ]
1320  1  29:5  375            IF DKVID = SOURCEVID THEN
1321  1  29:6  384              DKVID := GVID;                [ PREFIXED VOLUME'S NAME WAS CHANGED ]
1322  1  29:5  391            PRINTMESS(SOURCEVID,'',GVID)  [ TELL USER YOU DID THE CHANGE       ]
1323  1  29:4  398          END
1324  1  29:3  400        ELSE
1325  1  29:4  402          MESSAGES(ILLCHANGE,TRUE)      [ CAN'T CHANGE A VOLNAME TO A FILENAME ]
1326  1  29:1  406    UNTIL INSTRING = ''
1327  1  29:0  411  END [CHANGER] ;
1328  1  29:0  446
1329  1  29:0  446  [ ALLOWS THE USER TO REMOVE ANY SELECTED FILE FROM THE DIRECTORY ]
1330  1  30:D    1  PROCEDURE REMOVER;
1331  1  30:D    1  VAR
1332  1  30:D    1    DELETIONS : BITMAP;
1333  1  30:D    7    LINE,LOC : INTEGER;
1334  1  30:D    9
1335  1  30:0    0  BEGIN [ REMOVER ]
1336  1  30:1    0    REPEAT
1337  1  30:2    0      CHECKFILE('REMOVE','',1,FILEEXP,TRUE,TRUE,[OKFILE]);
1338  1  30:2   26      LINE := 0;    [ KEEPS TRACK OF WHAT LINE OF OUTPUT YOUR AT ]
1339  1  30:2   29      LOC := 0;
1340  1  30:2   32      FILLCHAR(DELETIONS,SIZEOF(DELETIONS),CHR(0)); [ INIT'S BITMAP ]
1341  1  30:2   39      WHILE SEARCHDIR('REMOVE',LOC,FALSE,TRUE) DO [ GET FILENAME ]
1342  1  30:3   58        BEGIN
1343  1  30:4   58          IF NOT QUESTION THEN
1344  1  30:5   62            BEGIN
1345  1  30:6   62              PRINTMESS(GVID,GDIR^[LOC].DTID,'REMOVED');
1346  1  30:6   82              IF SYSCOM^.CRTINFO.HEIGHT = LINE THEN  [ DON'T SCROLL OUTPUT ]
1347  1  30:7   91                BEGIN
1348  1  30:8   91                  NSPACEWAIT(FALSE);
```

```
1349  1  30:8   94                    CLEARSCREEN;
1350  1  30:8   97                    LINE := 0
1351  1  30:7   97                  END;
1352  1  30:6  100                LINE := LINE+1;
1353  1  30:5  105              END;
1354  1  30:4  105            DELETIONS.ENTRIES := DELETIONS.ENTRIES + 1; [ FILE TO BE REMOVED   ]
1355  1  30:4  117            DELETIONS.DIRENTRY [LOC] := TRUE  [ TOTAL # OF FILES TO BE REMOVED ]
1356  1  30:3  123          END;
1357  1  30:2  127        IF (FOUND IN [FILEFOUND,ABORTIT]) AND (DELETIONS.ENTRIES > 0) THEN
1358  1  30:3  142          BEGIN
1359  1  30:4  142            CLEARLINE;
1360  1  30:4  145            WRITE('UPDATE DIRECTORY ? '); [ MAKE USER CONFIRM THE REMOVAL ! ]
1361  1  30:4  174            IF NGETCHAR(TRUE) THEN
1362  1  30:5  181              ZAPENTRIES(DELETIONS,TRUE); [ WILL REMOVE THE SELECTED FILES  ]
1363  1  30:3  186          END;
1364  1  30:1  186      UNTIL INSTRING = ''
1365  1  30:0  189    END [REMOVER] ;
1366  1  30:0  212
1367  1  30:0  212    [ ALLOWS THE USER TO TRANSFER ANY FILE IN THE DIRECTORY TO ANOTHER DISK OR ]
1368  1  30:0  212    [ TO ANOTHER FILE. WILL ALSO PERFORM COMPLETE OR PARTIAL BINARY TRANSFERS  ]
1369  1  30:0  212    [ OF ONE DISK TO ANOTHER                                                   ]
1370  1  31:D    1    PROCEDURE TRANSFER;
1371  1  31:D    1    VAR
1372  1  31:D    1      LASTBLK,LOC : INTEGER;
1373  1  31:D    3
1374  1  31:D    3      [ PERFORMS THE ACTUAL TRANSFER OF THE FILE FROM ONE LOCATION TO ANOTHER ]
1375  1  32:D    1      PROCEDURE MOVEFILE;
1376  1  32:D    1      VAR
1377  1  32:D    1        RELBLK,NUMBLKS,NBLOCKS : INTEGER;
1378  1  32:D    4        FIRSTCALL,SINGLEDRIVE : BOOLEAN;
1379  1  32:D    6        GFIBP : FIBP;
1380  1  32:D    7        GFIB : UNTYPED;
1381  1  32:D   47
1382  1  32:0    0      BEGIN
1383  1  32:1    0        RESET(GFIB,FROMWHERE); [ OPEN SOURCE FILE ]
1384  1  32:1   22        CHECKRSLT(IORESULT);
1385  1  32:1   26        GFIBP := GETPTR(GFIB); [ GETS A POINTER TO THE HEADER OF THE SOURCE FILE ]
1386  1  32:1   34        RELBLK := 0;           [ BLOCK RELATIVE TO THE SOURCE FILE ]
1387  1  32:1   37        FIRSTCALL := TRUE;
1388  1  32:1   40        REPEAT
1389  1  32:2   40          NUMBLKS := LASTBLK - RELBLK;   [ BLOCKS LEFT TO TRANSFER ]
```

```
1390   1    32:2     47    IF NUMBLKS > GBUFBLKS THEN  [ GBUFBLKS = # OF BLKS IN TRANSFER BUFFER ]
1391   1    32:3     52      NUMBLKS := GBUFBLKS; [ UNABLE TO FIT WHOLE FILE IN TRANSFER BUFFER  ]
1392   1    32:2     55    NBLOCKS := BLOCKREAD(GFIB,GBUF^,NUMBLKS,RELBLK);
1393   1    32:2     70    CHECKRSLT(IORESULT);   [ NBLOCKS = # OF BLOCKS ACTUALLY READ ]
1394   1    32:2     74    IF FIRSTCALL THEN
1395   1    32:3     77      BEGIN
1396   1    32:4     77        FIRSTCALL := FALSE;
1397   1    32:4     80        SCANINPUT(TOWHERE,[NOVOL,BADDIR,BADFILE,UNBLKDVOL,OKDIR,OKFILE],
1398   1    32:4     89                                        FILEVOLEXP,DESTINATION,TRUE);
1399   1    32:4     96        IF ((GVID2 <> '') AND (GVID2 [1] = '#') AND (GUNIT = SOURCEUNIT)
1400   1    32:4    113                AND UNITABLE [GUNIT].UISBLKD) OR (GUNIT = 0) THEN
1401   1    32:4    128
1402   1    32:4    128          [ DESTINATION DISK IS NOT ON-LINE AT THE MOMENT ]
1403   1    32:5    128          BEGIN
1404   1    32:6    128            CLEARSCREEN;
1405   1    32:6    131            IF GUNIT = 0 THEN
1406   1    32:7    136              WRITELN('PUT IN ',GVID,':')
1407   1    32:6    176            ELSE
1408   1    32:7    178              WRITELN('INSERT DESTINATION DISK');
1409   1    32:6    217            NSPACEWAIT(TRUE);
1410   1    32:6    220
1411   1    32:6    220            [ MAKE SURE THAT THE USER PUT THE VOLUME ON-LINE ]
1412   1    32:6    220            SCANINPUT(TOWHERE,[BADFILE,OKFILE,BADDIR,OKDIR,UNBLKDVOL],
1413   1    32:6    229                                      FILEVOLEXP,DESTINATION,TRUE);
1414   1    32:5    236        END;
1415   1    32:4    236        IF GUNIT IN [1,2] THEN
1416   1    32:5    242          CLEARSCREEN;      [ DESTINATION IS THE CONSOLE: ]
1417   1    32:4    245        IF NOT FINDSAME(FALSE) THEN
1418   1    32:5    253          EXIT(MOVEFILE); [ USER DOESN'T WISH TO REMOVE THE DUPLICATE FILE ]
1419   1    32:4    257        IF NBLOCKS > DIRBLK THEN
1420   1    32:5    262          RISKVOLUME;      [ MAKE SURE THAT A DISK ISN'T INDANGERED ]
1421   1    32:4    264        REWRITE(LFIB,TOWHERE); [ OPEN DESTINATION FILE ]
1422   1    32:4    274        CHECKRSLT(IORESULT);
1423   1    32:4    278
1424   1    32:4    278        [ GET A POINTER TO THE HEADER OF THE DESTINATION FILE ]
1425   1    32:4    278        LFIBP := GETPTR(LFIB);
1426   1    32:4    286
1427   1    32:4    286        IF NOT LFIBP^.FISBLKD AND GFIBP^.FISBLKD AND
1428   1    32:4    292                                  (GFIBP^.FHEADER.DFKIND = TEXTFILE) THEN
1429   1    32:5    303          BEGIN [ DISK TO CHARACTER DEVICE DON'T TRANSFER HEADING ]
1430   1    32:6    303            NBLOCKS := NBLOCKS-2;
```

```
1431   1    32:6    308                MOVELEFT(GBUF^[FBLKSIZE+FBLKSIZE],GBUF^,NBLOCKS*FBLKSIZE)
1432   1    32:5    327              END;
1433   1    32:4    327            SINGLEDRIVE := (LFIBP^.FVID <> GFIBP^.FVID) AND
1434   1    32:4    335                                              (LFIBP^.FUNIT = GFIBP^.FUNIT)
1435   1    32:3    340          END
1436   1    32:2    343        ELSE
1437   1    32:3    345          IF SINGLEDRIVE THEN [ ALLOW USER TO INSERT DESTINATION DISK ]
1438   1    32:4    348            INSERTVOLUME(LFIBP^.FUNIT,LFIBP^.FVID,FALSE);
1439   1    32:2    356          NUMBLKS := BLOCKWRITE(LFIB,GBUF^,NBLOCKS,RELBLK);
1440   1    32:2    371          CHECKRSLT(IORESULT);
1441   1    32:2    375          IF NBLOCKS <> NUMBLKS THEN
1442   1    32:3    380            MESSAGES(FILEFULL,TRUE);   [ WASN'T ABLE TO WRITE OUT ALL THE BLOCKS ]
1443   1    32:2    386          IF SINGLEDRIVE AND NOT EOF(GFIB) THEN
1444   1    32:2    398             [ ALLOW USER TO INSERT SOURCE DISK ]
1445   1    32:3    398             INSERTVOLUME(LFIBP^.FUNIT,GFIBP^.FVID,FALSE);
1446   1    32:2    406          RELBLK := RELBLK + NUMBLKS   [ INCREMENT RELATIVE BLOCK TO THE FILE ]
1447   1    32:1    407        UNTIL (RELBLK = LASTBLK) OR EOF(GFIB);
1448   1    32:1    426        WITH LFIBP^,GFIBP^,FHEADER DO
1449   1    32:2    434          BEGIN  [ MAKE THE HEADERS TO THE TWO FILES THE SAME ]
1450   1    32:3    434            FHEADER.DLASTBYTE := DLASTBYTE;
1451   1    32:3    443            FHEADER.DFKIND := DFKIND;
1452   1    32:3    457            FHEADER.DACCESS := DACCESS;
1453   1    32:3    467            IF (DACCESS.MONTH = 0) AND (THEDATE.MONTH > 0) THEN
1454   1    32:4    487              FHEADER.DACCESS := THEDATE
1455   1    32:2    491          END;
1456   1    32:1    496        CLOSE(LFIB,LOCK);
1457   1    32:1    502        CHECKRSLT(IORESULT);
1458   1    32:1    506        PRINTMESS(GFIBP^.FVID,GFIBP^.FHEADER.DTID,
1459   1    32:1    512                          CONCAT(LFIBP^.FVID,':',LFIBP^.FHEADER.DTID));
1460   1    32:1    547        CLOSE(GFIB);
1461   1    32:0    553      END;
1462   1    32:0    576
1463   1    31:0      0    BEGIN [ TRANSFER ]
1464   1    31:1      0      REPEAT
1465   1    31:2      0        CHECKFILE('TRANSFER','TO WHERE',0,FILEVOLEXP,TRUE,TRUE,
1466   1    31:2     28                                  [BADDIR,OKFILE,OKDIR,UNBLKDVOL]);
1467   1    31:2     36        LASTBLK := MAXINT;   [ WILL BE SET TO THE # OF BLOCKS TO TRANSFER ]
1468   1    31:2     41        LOC := 0;
1469   1    31:2     44        IF (STRING1 = '') AND NOT WILDCARD THEN
1470   1    31:3     57          BEGIN     [ DISK TO DISK BINARY TRANSFER ]
1471   1    31:4     57            IF LASTSTATE IN [OKDIR,BADDIR] THEN
```

```
1472  1  31:5   63            GETBLOCKS('TRANSFER','BLOCKS','# OF BLOCKS TO TRANSFER',1,LASTBLK);
1473  1  31:4  114          MOVEFILE
1474  1  31:3  114        END
1475  1  31:2  116      ELSE
1476  1  31:3  116        WHILE SEARCHDIR('TRANSFER',LOC,TRUE,TRUE) DO
1477  1  31:4  139          MOVEFILE
1478  1  31:1  139    UNTIL INSTRING = '';
1479  1  31:0  153  END [ TRANSFER ];
1480  1  31:0  170
1481  1  31:0  170  [ ALLOWS THE USER TO CREATE FILE(S) ON THE DISK ]
1482  1  33:D    1  PROCEDURE MAKEFILE;
1483  1  33:D    1  VAR
1484  1  33:D    1  GFIB : UNTYPED;
1485  1  33:D   41  GFIBP : FIBP;
1486  1  33:D   42
1487  1  33:0    0  BEGIN [ MAKEFILE ]
1488  1  33:1    0    REPEAT
1489  1  33:2   12      CHECKFILE('MAKE','',1,FILEEXP,FALSE,TRUE,[BADFILE,OKFILE]);
1490  1  33:2   36      IF FINDSAME(TRUE) THEN       [ CHECK FOR AN EXISTING FILE WITH THIS NAME ]
1491  1  33:3   43        BEGIN
1492  1  33:4   43          REWRITE(GFIB,FROMWHERE); [ OPEN THE FILE ]
1493  1  33:4   53          CHECKRSLT(IORESULT);
1494  1  33:4   57          GFIBP := GETPTR(GFIB);   [ GET A POINTER TO THE HEADER OF THE FILE ]
1495  1  33:4   65          WITH GFIBP^ DO  [ NEED TO BE SURE THE FILE IS OF THE CORRECT SIZE ]
1496  1  33:5   69            FMAXBLK := FHEADER.DLASTBLK-FHEADER.DFIRSTBLK;
1497  1  33:4   83          CLOSE(GFIB,LOCK);
1498  1  33:4   89          CHECKRSLT(IORESULT);
1499  1  33:4   93          CLEARLINE;
1500  1  33:4   96          WRITELN(GVID,':',GTID,' MADE')
1501  1  33:3  143        END
1502  1  33:1  143    UNTIL INSTRING = ''
1503  1  33:0  146  END [MAKEFILE] ;
1504  1  33:0  174
1505  1  33:0  174
1506  1  33:0  174 [$I FILER.C.TEXT]
1506  1  33:0  174 [$I FILER.D.TEXT]
1507  1  33:0  174 [      COPYRIGHT (C) 1979, REGENTS OF THE UNIVERSITY OF CALIFORNIA.      ]
1508  1  33:0  174 [      PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-       ]
1509  1  33:0  174 [      TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE      ]
1510  1  33:0  174 [      OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.             ]
1511  1  33:0  174
```

```
1512  1   33:0   174 [---------------------- WORKFILE MAINTANENCE PROCEDURES -----------------------]
1513  1   33:0   174
1514  1   33:0   174   [ ALLOWS THE USER TO SAVE HIS WORKFILE UNDER ANY DESIRED NAME ]
1515  1   34:0     3   FUNCTION SAVEWORK : BOOLEAN;
1516  1   34:0     3   TYPE
1517  1   34:0     3     FILESTRG = STRING [4];
1518  1   34:0     3   VAR
1519  1   34:0     3     GS : SHORTSTRING;
1520  1   34:0    16     OK : BOOLEAN;
1521  1   34:0    17
1522  1   35:0     3     FUNCTION SAVEIT(WHATFILE : FILESTRG; WHICH : FILEKIND;
1523  1   35:0     5         VAR TITLE : TID; VAR SAVED,GOTIT  : BOOLEAN; MSG : INTEGER) : BOOLEAN;
1524  1   35:0    12     VAR
1525  1   35:0    12       GFIB : UNTYPED;
1526  1   35:0    52       GFIBP : FIBP;
1527  1   35:0    53
1528  1   35:0    53     [ CHANGES THE NAME OF THE WORKFILE TO THE NAME DESIRED BY THE USER ]
1529  1   35:0     0     BEGIN [ SAVEIT ]
1530  1   35:1     0       SAVED := TRUE;
1531  1   35:1    20       WITH USERINFO DO
1532  1   35:2    20         BEGIN
1533  1   35:3    20           RESET(GFIB,CONCAT('*SYSTEM.WRK.',WHATFILE));
1534  1   35:3    61           GFIBP := GETPTR(GFIB);
1535  1   35:3    69           WITH USERINFO, GFIBP^.FHEADER DO
1536  1   35:4    75           IF GFIBP^.FISOPEN THEN
1537  1   35:5    80             BEGIN
1538  1   35:6    80               DACCESS.YEAR := 100;
1539  1   35:6    88               TITLE := CONCAT(WORKTID,'.',WHATFILE);[ CHANGE THE WORKFILE NAME ]
1540  1   35:6   123               DTID := TITLE;    [ CHANGE THE NAME OF THE FILE ]
1541  1   35:6   130               SAVEIT := TRUE;
1542  1   35:6   133               CLOSE(GFIB,NORMAL)
1543  1   35:5   139             END
1544  1   35:4   139           ELSE
1545  1   35:5   141             BEGIN
1546  1   35:6   141               SAVEIT := FALSE;
1547  1   35:6   144               GOTIT := FALSE;
1548  1   35:6   147               MESSAGES(MSG,FALSE)  [ COULDN'T FIND THE WORKFILE ]
1549  1   35:5   149             END
1550  1   35:2   151         END
1551  1   35:0   151     END [ SAVEIT ];
1552  1   35:0   170
```

```
1553   1   35:0   170     [ CREATE INPUT STRING TO BE SENT TO THE T(RANSFER COMMAND ]
1554   1   36:0     1     PROCEDURE CONCATIT(STR : FILESTRG; SAVED : BOOLEAN);
1555   1   36:0     0     BEGIN
1556   1   36:1     3       IF NOT SAVED THEN
1557   1   36:2     9         INSTRING := CONCAT(INSTRING,'*SYSTEM.WRK.',STR,',',
1558   1   36:2    73                                          GVID2,':',GTID,'.',STR,',')
1559   1   36:0   139     END;
1560   1   36:0   156
1561   1   34:0     0   BEGIN [ SAVEWORK ]
1562   1   34:1     0     WITH USERINFO DO
1563   1   34:2     3       BEGIN
1564   1   34:3     0         SAVEWORK := FALSE; [ WILL BE SET TO TRUE IF SAVING TO A DIFFERENT DISK ]
1565   1   34:3     3         GTID := WORKTID;
1566   1   34:3    10         GVID := WORKVID;
1567   1   34:3    17         GVID2 := WORKVID;
1568   1   34:3    24         IF TEXTSAVED AND CODESAVED THEN   [ ERROR NOTHING TO SAVE ]
1569   1   34:4    29           BEGIN
1570   1   34:5    29             WRITELN;
1571   1   34:5    35             IF GOTSYM OR GOTCODE THEN
1572   1   34:6    44               MESSAGES(WRKSAVED,TRUE);    [ WORKFILE ALREADY SAVED ]
1573   1   34:5    50             MESSAGES(NOWRK,TRUE);         [ NO WORKFILE TO SAVE ]
1574   1   34:4    56           END;
1575   1   34:3    56         OK := FALSE;
1576   1   34:3    59         IF WORKTID <> '' THEN [ ALREADY HAVE A FILENAME ]
1577   1   34:4    69           BEGIN
1578   1   34:5    69             CLEARLINE;
1579   1   34:5    72             WRITE('SAVE AS ',WORKVID,':',WORKTID,' ? ');
1580   1   34:5   131             OK := NGETCHAR(FALSE)
1581   1   34:4   132           END;
1582   1   34:3   138         IF NOT OK THEN          [ NEED A NEW FILENAME ]
1583   1   34:4   142           CHECKFILE('SAVE AS','',1,FILEEXP,FALSE,TRUE,
1584   1   34:4   161                                  [NOVOL,BADDIR,BADFILE,OKDIR,OKFILE]);
1585   1   34:3   169         IF LENGTH(GTID) > TIDLENG-5 THEN
1586   1   34:4   179           MESSAGES(ILLFILEVOL,TRUE);   [ FILENAME IS TOO LONG ]
1587   1   34:3   185         IF GVID2 <> SYVID THEN
1588   1   34:4   194           BEGIN                          [ SAVE TO ALTERNATE DISK ]
1589   1   34:5   194             INSTRING := '';
1590   1   34:5   202             CONCATIT('TEXT',TEXTSAVED);
1591   1   34:5   212             CONCATIT('CODE',CODESAVED);
1592   1   34:5   222             DELETE(INSTRING,LENGTH(INSTRING),1);   [ REMOVE TRAILING COMMA ]
1593   1   34:5   234             SAVEWORK := TRUE;     [ WILL NEED TO ENTER T(RANSFER AFTER LEAVING ]
```

```
1594   1   34:5   237              EXIT(SAVEWORK)
1595   1   34:4   241            END;
1596   1   34:3   241          WORKTID := GTID;            [ CHANGE TITLE OF WORFILE ]
1597   1   34:3   248          WORKVID := GVID;            [ CHANGE VOLUME I.D. OF WORFILE ]
1598   1   34:3   255          CLEARLINE;
1599   1   34:3   256          IF NOT TEXTSAVED THEN
1600   1   34:4   262            BEGIN                     [ TEXT FILE NEEDS TO BE SAVED ]
1601   1   34:5   262              IF SAVEIT('TEXT',TEXTFILE,SYMTID,TEXTSAVED,GOTSYM,TEXTLOST) THEN
1602   1   34:6   287                BEGIN
1603   1   34:7   287                  IF CODESAVED THEN [ REMOVE OLD CODE EXISTING FILE ]
1604   1   34:8   290                    IF PURGEIT(CONCAT('*',WORKTID,'.CODE'),'') THEN
1605   1   34:9   337                      WRITE('OLD CODE REMOVED, ');
1606   1   34:7   365                  WRITE('TEXT FILE SAVED ')
1607   1   34:6   391                END;
1608   1   34:5   391              IF NOT CODESAVED THEN
1609   1   34:6   395                WRITE('& ')           [ WILL ALSO NEED TO SAVE NEW CODEFILE ]
1610   1   34:4   407            END;
1611   1   34:3   407          IF NOT CODESAVED THEN  [ SAVE CODE FILE ]
1612   1   34:4   411            IF SAVEIT('CODE',CODEFILE,CODETID,CODESAVED,GOTCODE,CODELOST) THEN
1613   1   34:5   436              WRITE('CODE FILE SAVED')
1614   1   34:2   461        END
1615   1   34:0   461    END [SAVEWORK] ;
1616   1   34:0   476
1617   1   34:0   476    [ INFORMS THE USER IF A WORKFILE EXISTS AND IF SO ]
1618   1   34:0   476    [ WHAT NAME IT IS ASSOCIATED WITH                 ]
1619   1   37:0     1    PROCEDURE WHATWORK;
1620   1   37:0     0    BEGIN
1621   1   37:1     0      WRITEANDCLEAR;
1622   1   37:1     2      WITH USERINFO DO
1623   1   37:2     2        IF GOTSYM OR GOTCODE THEN          [ THERE'S A CODE OR TEXT FILE LOADED ]
1624   1   37:3    11          BEGIN
1625   1   37:4    11            IF WORKTID = '' THEN
1626   1   37:5    21              WRITE('NOT NAMED')
1627   1   37:4    40            ELSE
1628   1   37:5    42              WRITE('WORKFILE IS ',WORKVID,':',WORKTID);
1629   1   37:4    92            IF NOT (TEXTSAVED AND CODESAVED) THEN
1630   1   37:5    98              WRITE(' (NOT SAVED)')
1631   1   37:3   120          END
1632   1   37:2   120        ELSE
1633   1   37:3   122          WRITE('NO WORKFILE')
1634   1   37:0   143    END [WHATWORK] ;
```

```
1635   1    37:0   156
1636   1    37:0   156   { CLEARS THE PRESENT WORKFILE. USED IN GETWORK & NEWWORK }
1637   1    38:0     1   PROCEDURE CLEARWORK;
1638   1    38:0     0   BEGIN
1639   1    38:1     0     WITH USERINFO DO
1640   1    38:2     0       BEGIN
1641   1    38:3     0         GOTSYM := FALSE;
1642   1    38:3     4         GOTCODE := FALSE;
1643   1    38:3     8         WORKTID := '';
1644   1    38:3    16         SYMTID := '';
1645   1    38:3    24         CODETID := ''
1646   1    38:2    27       END
1647   1    38:0    32   END;
1648   1    38:0    44
1649   1    38:0    44   { CLEARS THE PRESENT WORKFILE. IF A SYSTEM.WRK EXISTS WILL REMOVE IT }
1650   1    39:0     1   PROCEDURE NEWWORK(GIVEBLURB: BOOLEAN);
1651   1    39:0     2   VAR
1652   1    39:0     2     GFIB : FILE;
1653   1    39:0     0   BEGIN { NEWWORK }
1654   1    39:1     0     WITH USERINFO DO
1655   1    39:2    12       BEGIN
1656   1    39:3    12         IF NOT (TEXTSAVED AND CODESAVED) THEN
1657   1    39:4    18           BEGIN     { CURRENT WORKFILE HASN'T BEEN SAVED VERIFY ITS REMOVAL }
1658   1    39:5    18             CLEARLINE;
1659   1    39:5    21             WRITE('THROW AWAY CURRENT WORKFILE ? ');
1660   1    39:5    61             IF NOT NGETCHAR(FALSE) THEN
1661   1    39:6    69               EXIT(CALLPROC);
1662   1    39:4    73           END
1663   1    39:3    73         ELSE
1664   1    39:4    75           IF GIVEBLURB THEN
1665   1    39:5    78             WRITELN;  { WASN'T CALLED FROM GETWORK }
1666   1    39:5    84
1667   1    39:5    84         { REMOVE ALL WORKFILES }
1668   1    39:3    84         IF PURGEIT('*SYSTEM.WRK.TEXT','') THEN;
1669   1    39:3   112         IF PURGEIT('*SYSTEM.WRK.CODE','') THEN;
1670   1    39:3   140         IF PURGEIT('*SYSTEM.LST.TEXT','') THEN;
1671   1    39:3   168
1672   1    39:3   168         { CHECK FOR A .BACK FILE IN CASE USER HAS A LARGE FILE EDITOR }
1673   1    39:3   168         IF PURGEIT(CONCAT(WORKTID,'.BACK'),'REMOVE') THEN;
1674   1    39:3   211
1675   1    39:3   211         TEXTSAVED := TRUE;
```

```
1675    1    39:3    214          CODESAVED := TRUE;
1677    1    39:3    217          IF GIVESLURS THEN
1678    1    39:4    220            WITH USERINFO DO  [ INFORM THE USER OF THE STATUS OF THE WORKFILE ]
1679    1    39:5    220              BEGIN
1680    1    39:6    220                CLEARLINE;
1681    1    39:6    223                WRITE('WORKFILE CLEARED');
1682    1    39:6    249                CLEARWORK;
1683    1    39:5    251              END
1684    1    39:2    251        END
1685    1    39:0    251    END [ NEWWORK ];
1686    1    39:0    270
1687    1    39:0    270    [ ALLOWS THE USER TO LOAD A NEW FILE NAME INTO HIS WORKFILE ]
1688    1    40:0      1    PROCEDURE GETWORK;
1689    1    40:0      1    TYPE
1690    1    40:0      1      SHORT = STRING[5];
1691    1    40:0      1    VAR
1692    1    40:0      1      DONE,OK : BOOLEAN;
1693    1    40:0      3      X : INTEGER;
1694    1    40:0      4
1695    1    40:0      4      [ CHECKS TO SEE WHETHER OR NOT THE REQUESTED FILE TO BE LOADED EXISTS ]
1696    1    41:0      3      FUNCTION CHECKIT(SUFFIX,MESS:SHORT; VAR TITLE:TID; VAR VOLID:VID) : BOOLEAN;
1697    1    41:0      0      BEGIN
1698    1    41:1      0        WITH USERINFO DO
1699    1    41:2     10        BEGIN
1700    1    41:3     10          CHECKIT := FALSE;
1701    1    41:3     13          SCANINPUT(CONCAT(WORKVID,':',WORKTID,SUFFIX),[OKFILE],0,NEITHER,TRUE);
1702    1    41:3     65          IF LASTSTATE = OKFILE THEN
1703    1    41:4     70            BEGIN      [ THE REQUESTED FILE HAS BEEN FOUND ]
1704    1    41:5     70              CHECKIT := TRUE;
1705    1    41:5     73              DONE := TRUE;
1706    1    41:5     77              TITLE := CONCAT(WORKTID,SUFFIX);
1707    1    41:5    102              VOLID := WORKVID;
1708    1    41:5    108              IF GOTSYM THEN
1709    1    41:6    113                WRITE('& ');
1710    1    41:5    125              WRITE(MESS)
1711    1    41:4    134            END
1712    1    41:2    134        END
1713    1    41:0    134      END;
1714    1    41:0    146
1715    1    40:0      0    BEGIN [ GETWORK ]
1716    1    40:1      0      NEWWORK(FALSE);     [ CLEAR EXISTING WORKFILE ]
```

```
1717   1   40:1     3     CHECKFILE('GET','',1,FILEEXP,FALSE,TRUE,[BADFILE,OKFILE]);
1718   1   40:1    26     WITH USERINFO DO
1719   1   40:2    26     BEGIN
1720   1   40:3    26       CLEARWORK;  [ CLEARWORK HASN'T CLEARED WORKFILE YET IN CASE OF NUL INPUT ]
1721   1   40:3    26       WORKVID := GVID;
1722   1   40:3    35       WORKTID := GTID;
1723   1   40:3    42       X := LENGTH(WORKTID);
1724   1   40:3    49       OK := X <= TIDLENG-5;       [ CAN A '.TEXT' OR '.CODE' SUFFIX BE ADDED ]
1725   1   40:3    56       CLEARLINE;
1726   1   40:3    59       REPEAT
1727   1   40:4    59         DONE := NOT OK;
1728   1   40:4    63         IF DONE AND (X > 5) THEN [ '.TEXT' OR '.CODE' SUFFIX MAY ALREADY EXIST ]
1729   1   40:5    70           IF (COPY(WORKTID,X-4,5)='.TEXT') OR (COPY(WORKTID,X-4,5)='.CODE') THEN
1730   1   40:6   121             DELETE(WORKTID,X - 4,5); [ REMOVE '.TEXT' OR '.CODE' SUFFIX ]
1731   1   40:4   131         IF (LENGTH(WORKTID) <= TIDLENG-5) AND (WORKTID <> '') THEN
1732   1   40:5   151           BEGIN [ SEE IF FILE IS IN DIRECTORY ]
1733   1   40:6   151             GOTSYM := CHECKIT('.TEXT','TEXT ',SYMTID,SYMVID);
1734   1   40:6   180             GOTCODE := CHECKIT('.CODE','CODE ',CODETID,CODEVID)
1735   1   40:5   202           END;
1736   1   40:4   209         OK := FALSE
1737   1   40:3   209       UNTIL DONE;
1738   1   40:3   215       IF NOT (GOTSYM OR GOTCODE) THEN
1739   1   40:4   225         BEGIN  [ WASN'T ABLE TO FIND THE FILE ]
1740   1   40:5   225           CLEARWORK;
1741   1   40:5   227           WRITE('NO ')
1742   1   40:4   240         END;
1743   1   40:3   240       WRITE('FILE LOADED')
1744   1   40:2   261     END
1745   1   40:0   261   END [GETWORK] ;
1746   1   40:0   276
1747   1   40:0   276
1748   1   40:0   276 [$I FILER.D.TEXT]
1748   1   40:0   276 [$I FILER.E.TEXT]
1749   1   40:0   276 [     COPYRIGHT (C) 1979, REGENTS OF THE UNIVERSITY OF CALIFORNIA.     ]
1750   1   40:0   276 [     PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-       ]
1751   1   40:0   276 [     TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE      ]
1752   1   40:0   276 [     OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.             ]
1753   1   40:0   276
1754   1   40:0   276 [----------------------- DIRECTORY RELATED ROUTINES -------------------------]
1755   1   40:0   276
1756   1   40:0   276   [ ALLOWS THE USER TO SET THE DATE IN THE DIRECTORY ]
```

```
1757   1    42:0      1   PROCEDURE DATESET;
1758   1    42:0      1   CONST
1759   1    42:0      1     DASH = '-';      { ONLY DELIMMITER ALLOWED TO SEPARATE FIELDS IN THE DATE }
1760   1    42:0      1   VAR
1761   1    42:0      1     NUM : INTEGER;
1762   1    42:0      2
1763   1    42:0      2     { DELETES INPUT STRING UP TO THE NEXT FIELD DELIMITTER }
1764   1    43:0      1     PROCEDURE ZAPIT;
1765   1    43:0      0     BEGIN
1766   1    43:1      0       DELETE(INSTRING,1,SCAN(LENGTH(INSTRING),=DASH,INSTRING[1]));
1767   1    43:1     21       DELETE(INSTRING,1,1)
1768   1    43:0     29     END;
1769   1    43:0     42
1770   1    43:0     42     { TRANSLATES NUMBER IN CHARACTER REPRESENTAION INTO INTEGER REPRESENTATION }
1771   1    43:0     42     { AND CHECKS TO SEE IF IT IS IN THE ALLOWABLE RANGE                       }
1772   1    44:0      3     FUNCTION GETNUMBER(MAX : INTEGER) : BOOLEAN;
1773   1    44:0      4     VAR
1774   1    44:0      4       X,STOP : INTEGER;
1775   1    44:0      0     BEGIN
1776   1    44:1      0       NUM := 0;
1777   1    44:1      4       STOP := SCAN(LENGTH(INSTRING),=DASH,INSTRING[1]);
1778   1    44:1     20       FOR X := 1 TO STOP DO
1779   1    44:2     31         IF INSTRING[X] IN DIGITS THEN
1780   1    44:3     45           NUM := NUM*10+ORD(INSTRING[X])-ORD('0');
1781   1    44:1     68       GETNUMBER := (NUM > 0) AND (NUM <= MAX)
1782   1    44:0     78     END;
1783   1    44:0     96
1784   1    42:0      0   BEGIN {DATESET}
1785   1    42:1      0     WITH THEDATE DO
1786   1    42:2      0       BEGIN
1787   1    42:3      0         WRITELN('DATE SET: <1..31>-<JAN..DEC>-<00..99>');
1788   1    42:3     53         IF MONTH <> 0 THEN { WRITE OUT PRESENT DATE IF IT IS VALID }
1789   1    42:4     63           WRITELN('TODAY IS ',DAY,DASH,COPY(MONTHSTR,MONTH*3+1,3),DASH,YEAR);
1790   1    42:3    158         WRITE('NEW DATE ? ');
1791   1    42:3    179         READLN(INSTRING);
1792   1    42:3    197         EATSPACES(INSTRING);
1793   1    42:3    202         IF GETNUMBER(31) THEN
1794   1    42:4    209           DAY := NUM;    { A NEW DAY WAS FOUND }
1795   1    42:3    216         ZAPIT;  { DELETE INPUT STRING UP TO THE NEXT DELIMMITER }
1796   1    42:3    218         IF INSTRING[0] > CHR(2) THEN
1797   1    42:4    227           BEGIN
```

```
1798   1   42:5   227            GVID [0] := CHR(3);
1799   1   42:5   232            MOVELEFT(INSTRING[1],GVID[1],3);
1800   1   42:5   242            FOR NUM := 2 TO 3 DO
1801   1   42:6   255              IF (GVID [NUM] >= 'A') AND (GVID [NUM] <= 'Z') THEN
1802   1   42:7   268                GVID [NUM] := CHR( ORD(GVID [NUM]) - ORD('A') + ORD('A'));
1803   1   42:5   287            FOR NUM := 1 TO 12 DO
1804   1   42:6   298             IF COPY(MONTHSTR,NUM*3+1,3) = GVID THEN
1805   1   42:7   320               MONTH := NUM    [ A NEW VALID MONTH HAS BEEN GIVEN ]
1806   1   42:4   325          END;
1807   1   42:3   334        ZAPIT;  [ DELETE INPUT STRING UP TO THE NEXT DELIMMITER ]
1808   1   42:3   336        IF GETNUMBER(99) THEN
1809   1   42:4   343          YEAR := NUM;  [ A VALID YEAR HAS BEEN GIVEN ]
1810   1   42:3   350        SCANINPUT(CONCAT(SYVID,':'),[OKDIR],0,NEITHER,TRUE);
1811   1   42:3   383        IF (LASTSTATE = OKDIR) AND (GUNIT = SYSCOM^.SYSUNIT) THEN
1812   1   42:4   395          BEGIN [ THE ROOT VOLUME IS ON-LINE, WRITE THE DATE OUT TO ITS DIR. ]
1813   1   42:5   395            GDIR^[0].DLASTBOOT := THEDATE;
1814   1   42:5   406            WRITEDIR(GUNIT,GDIR);
1815   1   42:4   411          END;
1816   1   42:3   411        WRITE('THE DATE IS ',DAY,DASH,COPY(MONTHSTR,MONTH*3+1,3),DASH,YEAR)
1817   1   42:2   503    END
1818   1   42:0   503  END [ DATESET ] ;
1819   1   42:0   520
1820   1   42:0   520  [ ALLOWS THE USER TO SEE WHAT & WHERE HIS/HER FILES ARE IN THE DIRECTORY ]
1821   1   45:0     1  PROCEDURE LISTDIR(DETAIL: BOOLEAN);
1822   1   45:0     2  VAR
1823   1   45:0     2    NOFILES,ALTFILE : BOOLEAN;
1824   1   45:0     4    OUT : TEXT;
1825   1   45:0   305    LISTED,LOC,LINE,LARGEST,FREEBLKS,USEDAREA,USEDBLKS: INTEGER;
1826   1   45:0   312
1827   1   45:0   312    [ KEEPS TRACK OF WHAT LINE OF OUTPUT WE'RE AT SO WE DON'T SCROLL LISTSING ]
1828   1   46:0     1    PROCEDURE WRITELINE;
1829   1   46:0     0    BEGIN
1830   1   46:1     0      IF (LINE = SYSCOM^.CRTINFO.HEIGHT) OR (LINE = 0) THEN
1831   1   46:2    19        BEGIN   [ WRITE OUT VOLUME NAME AT TOP OF EACH PAGE OF OUTPUT ]
1832   1   46:3    19          HOMECURSOR;
1833   1   46:3    22          CLEARLINE;
1834   1   46:3    25          IF NOT ((LINE = 0) OR QUESTION) THEN
1835   1   46:4    36            NSPACEWAIT(FALSE);  [ LET USER LOOK AT WHATS ON THE SCREEN ]
1836   1   46:3    39          IF (NOT UNITABLE[DESTUNIT].UISBLKD) OR QUESTION THEN
1837   1   46:4    51            CLEARSCREEN;          [ LISTING TO CONSOLE ]
1838   1   46:3    54          WRITELN(OUT);
```

```
1839   1   46:3    50          WRITE(OUT,SOURCEVID,':');
1840   1   46:3    77          LINE := 2;
1841   1   46:3    82          IF ALTFILE THEN
1842   1   46:4    87            LINE := SYSCOM^.CRTINFO.HEIGHT; [ LIST THE HEADING ONLY ONCE ]
1843   1   46:2    96        END;
1844   1   46:1    96        WRITELN(OUT);
1845   1   46:1   102        LINE := LINE+1
1846   1   46:0   106      END [WRITELINE] ;
1847   1   46:0   124
1848   1   46:0   124      [ WRITES OUT UNUSED AREAS ON THE DISK ]
1849   1   47:0     1      PROCEDURE FREECHECK(FIRSTOPEN,NEXTUSED: INTEGER);
1850   1   47:0     3      VAR
1851   1   47:0     3        FREEAREA: INTEGER;
1852   1   47:0     4
1853   1   47:0     0      BEGIN
1854   1   47:1     0        FREEAREA := NEXTUSED-FIRSTOPEN; [ FINDS SPACE BETWEEN LAST & NEXT FILE  ]
1855   1   47:1     5        IF FREEAREA > LARGEST THEN
1856   1   47:2    13          LARGEST := FREEAREA;          [ IS THIS THE BIGGEST SPACE ON THE DISK ]
1857   1   47:1    18        FREEBLKS := FREEBLKS+FREEAREA;  [ RUNNING TOTAL OF FREE BLOCKS ]
1858   1   47:1    28        IF (FREEAREA > 0) AND DETAIL THEN [ EXTENDED LISTING ]
1859   1   47:2    37          BEGIN
1860   1   47:3    37            WRITE(OUT,'< UNUSED >',FREEAREA:10,'':10);
1861   1   47:3    75            IF FAST THEN
1862   1   47:4    78              WRITE(OUT,FIRSTOPEN:6)
1863   1   47:3    86            ELSE
1864   1   47:4    88              WRITE(OUT,FIRSTOPEN:5);
1865   1   47:3    96            WRITELINE
1866   1   47:2    96          END
1867   1   47:0    98      END [FREECHECK] ;
1868   1   47:0   110
1869   1   45:0     0    BEGIN [LISTDIR]
1870   1   45:1     0      CHECKFILE('DIR LISTING OF','',-1,FILEBLKDEXP,TRUE,FALSE,[OKDIR,OKFILE]);
1871   1   45:1    47      ALTFILE := TOWHERE <> '';   [ ARE WE LISTING TO CONSOLE: OR NOT ? ]
1872   1   45:1    57      IF ALTFILE THEN
1873   1   45:2    60        BEGIN
1874   1   45:3    60          SCANINPUT(TOWHERE,[BADFILE,OKFILE,UNBLKDVOL],FILEUNBLKDEXP,DESTINATION
1875   1   45:3    72                                        ,TRUE);
1876   1   45:3    76          SCANINPUT(FROMWHERE,[OKDIR,OKFILE],FILEBLKDEXP,SOURCE,TRUE)
1877   1   45:2    90        END
1878   1   45:1    92      ELSE
1879   1   45:2    94        TOWHERE := '#2:';           [ WE ARE LISTING TO THE CONSOLE:              ]
```

```
1880  1  45:1  105   REWRITE(OUT,TOWHERE);          { CHANGE OUTPUT TO APPROPRIATE DEVICE          }
1881  1  45:1  115   CHECKRSLT(IORESULT);
1882  1  45:1  119   LISTED := 0;                   { TOTAL # OF FILES LISTED                      }
1883  1  45:1  123   LOC := 0;
1884  1  45:1  127   LINE := 0;                     { OUTPUT LINE # TO AVOID SCROLLING OF LISTING }
1885  1  45:1  131   LARGEST := 0;                  { LARGEST FREE AREA ON THE DISK                }
1886  1  45:1  135   FREEBLKS := 0;                 { TOTAL # OF FREE BLOCKS ON THE DISK           }
1887  1  45:1  139   USEDAREA := 0;                 { SIZE OF FILE LISTED                          }
1888  1  45:1  143   USEDBLKS := GDIR^[0].DLASTBLK;  { TOTAL # OF BLOCKS BEING USED               }
1889  1  45:1  151   IF STRING1 = '' THEN
1890  1  45:2  161     BEGIN  { IN CASE OF EMPTY DIR THE # OF UNUSED BLOCKS WILL BE DISPLAYED }
1891  1  45:3  161       NOFILES := NOT WILDCARD AND (GDIR^ [0].DNUMFILES = 0);
1892  1  45:3  174       WILDCARD := TRUE;
1893  1  45:2  177     END;
1894  1  45:1  177   IF WILDCARD THEN
1895  1  45:2  180     WRITELINE   { CORRECTION FOR GOOD LOOKING OUTPUT }
1896  1  45:1  180   ELSE
1897  1  45:2  184     LINE := SYSCOM^.CRTINFO.HEIGHT+1; { SINGLE FILE REQ. DON'T WRITE HEADING }
1898  1  45:1  194   IF NOT NOFILES THEN
1899  1  45:2  198     WHILE SEARCHDIR('LIST',LOC,FALSE,FALSE) DO { GET FILE TO BE LISTED }
1900  1  45:3  216       BEGIN
1901  1  45:4  216       IF UNITABLE[DESTUNIT].UISBLKD AND (NOT QUESTION) THEN

1902  1  45:4  228
1903  1  45:4  228         { WRITING DIRECTORY OUT TO A BLOCKED DEVICE }
1904  1  45:5  228         IF LISTED = 0 THEN { FIRST CALL TO PROCEDURE }
1905  1  45:6  235           BEGIN
1906  1  45:7  235             WRITEANDCLEAR;
1907  1  45:7  237             WRITE('WRITING')
1908  1  45:6  254           END
1909  1  45:5  254         ELSE
1910  1  45:6  256           WRITE('.')
1911  1  45:4  264       ELSE
1912  1  45:5  266         CLEARLINE;
1913  1  45:4  269       LISTED := LISTED + 1;
1914  1  45:4  277       WITH GDIR^[LOC] DO
1915  1  45:5  286         BEGIN
1916  1  45:6  286           FREECHECK(GDIR^[LOC-1].DLASTBLK,DFIRSTBLK);{ CHECK FOR FREE BLOCKS }
1917  1  45:6  301           USEDAREA := DLASTBLK-DFIRSTBLK; { AREA USED }
1918  1  45:6  313           USEDBLKS := USEDBLKS+USEDAREA;  { RUNNING TOTAL OF USED BLOCKS }
1919  1  45:6  323           WRITE(OUT,DTID,'':TIDLENG-LENGTH(DTID)+1,USEDAREA:4,DACCESS.DAY:3
1920  1  45:6  372                ,'-',COPY(MONTHSTR,DACCESS.MONTH*3+1,3),'-',DACCESS.YEAR:2);
```

```
1921   1   45:6   435            IF DETAIL THEN                    [ EXTENDED LISTING ]
1922   1   45:7   438              IF FAST THEN
1923   1   45:8   441                WRITE(OUT,DFIRSTBLK:6,DLASTBYTE:6,'':2,
1924   1   45:8   471                                  COPY(TYPESTR,ORD(DFKIND)*4+1,4),'FILE')
1925   1   45:7   515              ELSE
1926   1   45:8   517                WRITE(OUT,DFIRSTBLK:5,' ',COPY(TYPESTR,ORD(DFKIND)*4+1,4));
1927   1   45:6   565              WRITELINE
1928   1   45:5   565            END
1929   1   45:3   567        END;
1930   1   45:1   569      IF (FOUND IN [FILEFOUND,ABORTIT]) OR NOFILES THEN
1931   1   45:2   578        BEGIN
1932   1   45:3   578          IF WILDCARD THEN
1933   1   45:4   531          BEGIN
1934   1   45:5   581            FREECHECK(GDIR^[LOC].DLASTBLK,GDIR^[0].DEOVBLK);
1935   1   45:5   595            WRITE(OUT,LISTED,'/',GDIR^[0].DNUMFILES,' FILES');
1936   1   45:5   638            IF FAST THEN
1937   1   45:6   641              WRITE(OUT,'<LISTED/IN-DIR>, ',USEDBLKS,' BLOCKS USED');
1938   1   45:5   697            WRITE(OUT,', ',FREEBLKS,' UNUSED',', ',LARGEST,' IN LARGEST');
1939   1   45:5   773            IF ALTFILE THEN
1940   1   45:6   776              WRITELN(OUT)
1941   1   45:4   781          END;
1942   1   45:3   781          CHECKRSLT(IORESULT);
1943   1   45:3   785          CLOSE(OUT,LOCK);
1944   1   45:3   791          CHECKRSLT(IORESULT);
1945   1   45:2   795        END
1946   1   45:0   795    END [LISTDIR] ;
1947   1   45:0   824
1948   1   45:0   824    [ LISTS THE VOLUMES THAT ARE ON-LINE ]
1949   1   48:0     1    PROCEDURE LISTVOLS;
1950   1   48:0     0    BEGIN
1951   1   48:1     0      WRITELN;
1952   1   48:1     6      WRITELN('VOLS ON-LINE:');
1953   1   48:1    35      GUNIT := VOLSEARCH(GVID,TRUE,GDIR); [ UPDATE UNITABLE ]
1954   1   48:1    47      FOR GUNIT := 1 TO MAXUNIT DO
1955   1   48:2    58       WITH UNITABLE[GUNIT] DO
1956   1   48:3    66        IF UVID <> '' THEN [ VOLUME IS ON-LINE ]
1957   1   48:4    74          BEGIN
1958   1   48:5    74            WRITE(GUNIT:3);
1959   1   48:5    82            IF UISBLKD THEN
1960   1   48:6    86              WRITE(' # ') [ BLOCKED DEVICE   ]
1961   1   48:5    99            ELSE
```

```
1962  1  48:6  101            WRITE('':3); [ UNBLOCKED DEVICE ]
1963  1  48:5  111          WRITELN(UVID,':')
1964  1  48:4  133        END;
1965  1  48:1  140    WRITELN('ROOT VOL IS - ',SYVID,':');    [ BOOTED VOLUME ]
1966  1  48:1  188    WRITELN('PREFIX IS   - ',DKVID,':')     [ PREFIXED VOLUME ]
1967  1  48:0  236  END;
1968  1  48:0  250
1969  1  48:3  250  [ CREATES AN EMPTY DIRECTORY ON A DISK ]
1970  1  49:0    1  PROCEDURE ZEROVOLUME;
1971  1  49:0    1  VAR
1972  1  49:0    1    LDE: DIRENTRY;
1973  1  49:0   14
1974  1  49:0    0  BEGIN [ ZEROVOLUME ]
1975  1  49:1    0    FILLCHAR(LDE,SIZEOF(LDE),CHR(0));
1976  1  49:1    7    LDE.DLASTBLK := DIRLASTBLK; [ LEAVE ROOM FOR DIRECTORY AND BOOTSTRAP ]
1977  1  49:1   10    CHECKFILE('ZERO DIR OF','',1,BLKDEXP,FALSE,FALSE,[OKDIR,BADDIR]);
1978  1  49:1   39    RISKVOLUME;     [ DOES THE DISK ALREADY HAVE A DIRECTORY ON IT ? ]
1979  1  49:1   41    WRITE('DUPLICATE DIR ? ');  [ DOES THE USER WANT A BACKUP DIRECTORY ? ]
1980  1  49:1   67    IF NGETCHAR(TRUE) THEN
1981  1  49:2   74      LDE.DLASTBLK := DUPDIRLASTBLK;
1982  1  49:1   77    GETBLOCKS('ARE THERE','BLKS ON THE DISK','# OF BLOCKS ON THE DISK',
1983  1  49:1  134                                              LDE.DLASTBLK,LDE.DEOVBLK);
1984  1  49:1  139    REPEAT
1985  1  49:2  139      WRITE('NEW VOL NAME ? ');
1986  1  49:2  164      READLN(INSTRING);
1987  1  49:2  182      EATSPACES(INSTRING);
1988  1  49:2  187      IF (INSTRING [LENGTH(INSTRING)] <> ':') AND (INSTRING <> '') THEN
1989  1  49:3  209        INSTRING := CONCAT(INSTRING,':');
1990  1  49:2  242      SCANINPUT(INSTRING,[NOVOL,OKDIR],BLKDEXP,NEITHER,TRUE);
1991  1  49:2  256      WRITE(GVID,': CORRECT ? ');
1992  1  49:1  287    UNTIL NGETCHAR(TRUE);
1993  1  49:1  294    WITH LDE DO
1994  1  49:2  294    BEGIN
1995  1  49:3  294      DFKIND := UNTYPEDFILE;  [ DIRECTORIES MUST BE OF THIS TYPE              ]
1996  1  49:3  300      DVID := GVID;           [ ENTERS NAME OF DIRECTORY                     ]
1997  1  49:3  306      DLASTBOOT := THEDATE;   [ USED TO SET THE SYSTEM DATE UPON BOOTING     ]
1998  1  49:3  313      INSERTVOLUME(SOURCEUNIT,SOURCEVID,TRUE);   [ DON'T KILL THE WRONG DISK ]
1999  1  49:3  319      UNITWRITE(SOURCEUNIT,LDE,SIZEOF(LDE),DIRBLK);
2000  1  49:3  328      CHECKRSLT(IORESULT);
2001  1  49:3  332      WRITE(DVID,': ZEROED')
2002  1  49:2  359    END
```

```
2003  1  49:0  359  END [ZEROVOLUME] ;
2004  1  49:0  374
2005  1  49:0  374 [---------------------- FILE MAINTANENCE PROCEDURES -------------------------]
2006  1  49:0  374
2007  1  49:0  374   [ INFORMS THE USER OF FILES ENDANGERED BY BAD BLOCKS ]
2008  1  50:0    1   PROCEDURE PRINTFILES;
2009  1  50:0    1   VAR
2010  1  50:0    1     I : INTEGER;
2011  1  50:0    0   BEGIN
2012  1  50:1    0     IF DIRMAP.ENTRIES > 0 THEN
2013  1  50:2    9       BEGIN    [ THERE ARE FILES ENDANGERED BY BAD BLOCKS ]
2014  1  50:3    9         WRITELN('FILE(S) ENDANGERED:');
2015  1  50:3   44         FOR I := 0 TO MAXDIR DO
2016  1  50:4   55           IF DIRMAP.DIRENTRY [I] THEN
2017  1  50:5   64             WITH GDIR^ [I] DO
2018  1  50:6   70               BEGIN
2019  1  50:7   70                 IF I = 0 THEN
2020  1  50:8   75                   WRITE('DIRECTORY','':7) [ THERE'S A BAD BLK IN THE DIRECTORY ]
2021  1  50:7  104                 ELSE
2022  1  50:8  106                   WRITE(DTID,'':16-LENGTH(DTID));   [ WRITE OUT THE FILES NAME ]
2023  1  50:7  132                 WRITELN(DFIRSTBLK:6,DLASTBLK:6)
2024  1  50:6  156               END
2025  1  50:2  156       END
2026  1  50:0  163   END;
2027  1  50:0  180
2028  1  50:0  180   [ DETERMINES WHAT FILE A BLOCK IS IN OR BETWEEN ]
2029  1  51:0    1   PROCEDURE WHICHFILE(VAR BADBLK:INTEGER; MARK : BOOLEAN);
2030  1  51:0    3   VAR
2031  1  51:0    3     X : INTEGER;
2032  1  51:0    0   BEGIN
2033  1  51:1    0     IF GDIR <> NIL THEN
2034  1  51:2    5       BEGIN
2035  1  51:3    5         FOR X := 0 TO GDIR^ [0].DNUMFILES DO
2036  1  51:4   21           IF GDIR^ [X].DLASTBLK > BADBLK THEN
2037  1  51:5   31             BEGIN              [ THE BLOCK MUST BE IN THIS FILE IF ANY AT ALL ]
2038  1  51:6   31               IF NOT MARK THEN
2039  1  51:7   35                 BADBLK := X   [ FOR K(RUNCH THIS IS ALL WE WANT TO KNOW   ]
2040  1  51:6   36               ELSE
2041  1  51:7   40                 IF GDIR^ [X].DFIRSTBLK <= BADBLK THEN
2042  1  51:8   50                   BEGIN       [ THE BLOCK IS IN THIS FILE MARK IT AS SUCH ]
2043  1  51:9   50                     DIRMAP.ENTRIES := DIRMAP.ENTRIES + 1;
```

```
2044   1   51:9    62                    DIRMAP.DIRENTRY [X] := TRUE
2045   1   51:8    68                  END;
2046   1   51:6    70                EXIT(WHICHFILE)
2047   1   51:5    74              END;
2048   1   51:3    81          IF NOT MARK THEN
2049   1   51:4    85            BADBLK := GDIR^ [0].DNUMFILES + 1    [ WELL NEED THIS FOR K(RUNCH ]
2050   1   51:2    92        END
2051   1   51:0    95    END;
2052   1   51:0   110
2053   1   51:0   110    [ SCANS THE BLOCKS ON A DISK FOR READ ERRORS ]
2054   1   52:0     1    PROCEDURE BADBLOCKS;
2055   1   52:0     1    VAR
2056   1   52:0     1      A : ABLOCK;
2057   1   52:0   257      BLK,TOTAL, NBLOCKS : INTEGER;
2058   1   52:0   260
2059   1   52:0     0    BEGIN [ BADBLOCKS ]
2060   1   52:1     0      CHECKFILE('BAD BLOCK SCAN OF','',1,BLKDEXP,FALSE,FALSE,[OKDIR,BADDIR]);
2061   1   52:1    35      GETBLOCKS('SCAN FOR','BLOCKS','SCAN FOR HOW MANY BLOCKS',1,NBLOCKS);
2062   1   52:1    88      TOTAL := 0;
2063   1   52:1    92      FOR BLK := 0 TO NBLOCKS-1 DO
2064   1   52:2   113        BEGIN
2065   1   52:3   113          UNITREAD(GUNIT,A,FBLKSIZE,BLK);
2066   1   52:3   126          IF IORESULT <> 0 THEN
2067   1   52:4   132            BEGIN                    [ AN ERROR WAS FOUND IN READING THE BLOCK   ]
2068   1   52:5   132              TOTAL := TOTAL + 1;
2069   1   52:5   140              WHICHFILE(BLK,TRUE);  [ WAS THE BAD BLOCK IN A FILE ?             ]
2070   1   52:5   146              WRITELN('BLOCK ',BLK,' IS BAD')
2071   1   52:4   195            END
2072   1   52:2   195        END;
2073   1   52:1   205      WRITELN(TOTAL,' BAD BLOCKS'); [ PRINT OUT THE FILES WITH BAD BLKS IN THEM ]
2074   1   52:1   242      PRINTFILES
2075   1   52:0   242    END [BADBLOCKS] ;
2076   1   52:0   258
2077   1   52:0   258    [ COMPARES SUCCESSIVE READS & WRITES FOR EQUALITY. IF THEY ARE EQUIVELENT ]
2078   1   52:0   258    [ DECLARES THAT THE BLOCK IS O.K. OTHERWISE DECLARES THE BLOCK AS BEING   ]
2079   1   52:0   258    [ BAD AND ALLOWS THE USER TO MARK THE AFFECTED BLOCKS AS SUCH             ]
2080   1   53:0     1    PROCEDURE XBLOCKS;
2081   1   53:0     1    VAR
2082   1   53:0     1      NEWDIR : ^INTEGER;
2083   1   53:0     2      BAD : ARRAY [0..HALFMAXDIR] OF RECORD
2084   1   53:0     2              FIRST,LAST : INTEGER
```

```
2085   1   53:0     2          END;
2086   1   53:0    82    BLK,LASTBLOCK,LOC,FIRSTBLK,LASTBLK : INTEGER;
2087   1   53:0    87    LOE : DIRENTRY;
2088   1   53:0   100    A,B : ABLOCK;
2089   1   53:0   612
2090   1   53:0     0  BEGIN [ XBLOCKS ]
2091   1   53:1     0    CHECKFILE('EXAMINE BLOCKS ON',',',1,BLKDEXP,FALSE,FALSE,[OKDIR,BADDIR]);
2092   1   53:1    35    CLEARLINE;
2093   1   53:1    38    WRITE('BLOCK-RANGE ? ');
2094   1   53:1    62    READ(FIRSTBLK);
2095   1   53:1    70    IF EOLN THEN
2096   1   53:2    80      LASTBLK := FIRSTBLK
2097   1   53:1    80    ELSE
2098   1   53:2    86      BEGIN
2099   1   53:3    86        READ(LASTBLK);
2100   1   53:3    94        IF NOT EOLN THEN
2101   1   53:4   105          WRITELN;
2102   1   53:3   111        LASTBLK := ABS(LASTBLK)
2103   1   53:2   114      END;
2104   1   53:1   116    IF GDIR <> NIL THEN
2105   1   53:2   121      IF LASTBLK >= GDIR^ [0].DEOVBLK THEN
2106   1   53:3   131        LASTBLK := GDIR^ [0].DEOVBLK-1;  [ DON'T WANT TO SEEK PAST END OF DISK ]
2107   1   53:1   140    IF (FIRSTBLK < 0) OR (FIRSTBLK > LASTBLK) THEN
2108   1   53:2   152      EXIT(XBLOCKS);                    [ INVALID BLOCK RANGE                 ]
2109   1   53:1   156    CLEARSCREEN;
2110   1   53:1   159    WRITELN;
2111   1   53:1   165    FOR BLK := FIRSTBLK TO LASTBLK DO
2112   1   53:2   182      WHICHFILE(BLK,TRUE);      [ DETERMINE WHAT FILES ARE IN THE BLOCK-RANGE ]
2113   1   53:1   195    IF DIRMAP.ENTRIES > 0 THEN
2114   1   53:2   204      BEGIN
2115   1   53:3   204        PRINTFILES;             [ PRINT WHAT FILES ARE INDANGERED ]
2116   1   53:3   206        WRITE('FIX THEM ? ');
2117   1   53:3   227        IF NOT NGETCHAR(TRUE) THEN
2118   1   53:4   235          EXIT(CALLPROC)
2119   1   53:2   239      END;
2120   1   53:1   239    FILLCHAR(BAD,SIZEOF(BAD),0);
2121   1   53:1   248    FILLCHAR(DIRMAP,SIZEOF(DIRMAP),0);
2122   1   53:1   255    LOC := 0;
2123   1   53:1   258    LASTBLOCK := -10;
2124   1   53:1   262    FOR BLK := FIRSTBLK TO LASTBLK DO
2125   1   53:2   279      BEGIN
```

```
2126  1  53:3  279          WRITE('BLOCK ',BLK);
2127  1  53:3  304          UNITREAD(GUNIT,A,FBLKSIZE,BLK);
2128  1  53:3  317          UNITWRITE(GUNIT,A,FBLKSIZE,BLK);
2129  1  53:3  330          IF IORESULT = 0 THEN
2130  1  53:4  336            UNITREAD(GUNIT,B,FBLKSIZE,BLK);
2131  1  53:3  348          IF (IORESULT = 0) AND (A = B) THEN
2132  1  53:4  364            WRITELN(' MAY BE OK')
2133  1  53:3  390          ELSE
2134  1  53:4  392            BEGIN
2135  1  53:5  392              WRITELN(' IS BAD');
2136  1  53:5  415              WHICHFILE(BLK,TRUE);   [ IS THE BAD BLOCK IN A FILE ? ]
2137  1  53:5  420              IF GDIR <> NIL THEN
2138  1  53:6  425                IF BLK > GDIR^ [0].DLASTBLK THEN
2139  1  53:7  435                  BEGIN  [ CALCULATE THE # OF BAD.XXXX.BAD FILES & WHERE THEY GO ]
2140  1  53:8  435                    IF LASTBLOCK+1 <> BLK THEN
2141  1  53:9  444                      BEGIN
2142  1  53:0  444                        BAD [0].FIRST := BAD [0].FIRST + 1;
2143  1  53:0  460                        LOC := LOC + 1;
2144  1  53:0  466                        IF LOC > (MAXDIR+DIRMAP.ENTRIES-GDIR^ [0].DNUMFILES) THEN
2145  1  53:1  485                          CHECKRSLT(ORD(INOROOM)); [ NO ROOM TO ADD BAD.XXXX.BAD ]
2146  1  53:0  488                        BAD [LOC].FIRST := BLK
2147  1  53:9  496                      END;
2148  1  53:8  499                    BAD [LOC].LAST := BLK;
2149  1  53:8  508                    LASTBLOCK := BLK
2150  1  53:7  508                  END
2151  1  53:4  512              END
2152  1  53:2  512          END;
2153  1  53:1  520        IF BAD [0].FIRST = 0 THEN
2154  1  53:2  530          EXIT(XBLOCKS);
2155  1  53:1  534        PRINTFILES;  [ WRITE OUT FILES THAT WILL BE REMOVED IF DIRECTORY IS MARKED ]
2156  1  53:1  536        WRITE('MARK BAD BLOCKS  ?');
2157  1  53:1  564        IF DIRMAP.ENTRIES > 0 THEN
2158  1  53:2  573          WRITE(' (FILES WILL BE REMOVED !)');
2159  1  53:1  609        WRITE(' (Y/N) ');
2160  1  53:1  626        IF NOT NGETCHAR(TRUE) THEN
2161  1  53:2  634          EXIT(CALLPROC);
2162  1  53:1  638        ZAPENTRIES(DIRMAP,FALSE);  [ REMOVE FILES WITH BAD BLOCKS INSIDE OF THEM ]
2163  1  53:1  643        WITH LDE DO
2164  1  53:2  643          BEGIN
2165  1  53:3  643            DFKIND := XDSKFILE;
2166  1  53:3  649            DLASTBYTE := FBLKSIZE;
```

```
2167  1  53:3  654          DACCESS := THEDATE;
2168  1  53:3  661          DTID := 'BAD.XXXXX.BAD'
2169  1  53:2  663        END;
2170  1  53:1  681      FOR BLK := 1 TO BAD [0].FIRST DO
2171  1  53:2  701        WITH LDE,BAD[BLK] DO
2172  1  53:3  710          BEGIN
2173  1  53:4  710            DFIRSTBLK := FIRST;
2174  1  53:4  716            DLASTBLK := LAST+1;
2175  1  53:4  724            FOR LOC := 4 DOWNTO 0 DO
2176  1  53:5  739              BEGIN   [ MAKES THE STARTING BLOCK # PART OF THE FILE NAME ]
2177  1  53:6  739                DTID[9-LOC] := CHR(FIRST DIV IPOT[LOC] + ORD('0'));
2178  1  53:6  761                FIRST := FIRST MOD IPOT[LOC]
2179  1  53:5  777              END;
2180  1  53:4  788            LOC := GDIR^[0].DNUMFILES;
2181  1  53:4  796            WHILE DFIRSTBLK < GDIR^[LOC].DLASTBLK DO
2182  1  53:5  807              LOC := LOC - 1;
2183  1  53:4  815            INSENTRY(LDE,LOC+1,GDIR);   [ ADD THE BAD.XXXX.BAD FILE ]
2184  1  53:3  825          END;
2185  1  53:1  833      UPDATEDIR;     [ WRITE OUT THE NEW DIRECTORY ]
2186  1  53:1  835      WRITE('BAD BLOCKS MARKED')
2187  1  53:0  862    END [XBLOCKS] ;
2188  1  53:0  888
2189  1  53:0  888    [ ALLOWS THE USER TO OPEN UP THE LARGEST FREE SPACE AVAILABLE ON THE ]
2190  1  53:0  888    [ DISK AT ANY DESIRED LOCATION                                      ]
2191  1  54:D    1    PROCEDURE KRUNCH;
2192  1  54:D    1    TYPE
2193  1  54:D    1      WAY = (FOURWARD,REVERSE);   [ DIRECTION FILES ARE BEING MOVED ]
2194  1  54:D    1
2195  1  54:D    1    VAR
2196  1  54:D    1      GINX,SPLIT,FIRSTBLK,NBLOCKS,RELBLOCK,CHUNKSIZE : INTEGER;
2197  1  54:D    7      REBOOT: BOOLEAN;
2198  1  54:D    8
2199  1  54:D    8      [ DOES THE CALCULATIONS FOR MOVING THE FILES ]
2200  1  55:D    1      PROCEDURE KRUNCHIT(DIRECTION : WAY; VAR STARTING,OTHER : INTEGER;
2201  1  55:D    4                                              STOPPING : INTEGER);
2202  1  55:D    5
2203  1  55:D    5        [ DOES THE ACTUAL MOVING OF THE FILES FROM ONE LOCATION TO THE NEXT ]
2204  1  56:D    1        PROCEDURE MOVEIT(INOUT : SHORTSTRING; BLOCK : INTEGER);
2205  1  56:D   16        VAR
2206  1  56:D   16          BLK,X,START,STOP : INTEGER;
2207  1  56:D   20
```

```
2208   1   56:0     0        BEGIN
2209   1   56:1     0          START := RELBLOCK;  [ RELATIVE BLOCK OF THE FILE TO START MOVING FROM ]
2210   1   56:1    10          STOP := START+CHUNKSIZE;  [ LAST REL. BLOCK IN THE FILE TO BE MOVED   ]
2211   1   56:1    18          IF DIRECTION = REVERSE THEN
2212   1   56:2    25            BEGIN      [ MUST NEGATE LOGIC PARAMS GIVEN ARE NEGATIVE & REVERSED ]
2213   1   56:3    25              START := -STOP;
2214   1   56:3    29              STOP := -RELBLOCK
2215   1   56:2    29            END;
2216   1   56:1    35          X := 0;
2217   1   56:1    38          WITH GDIR^[GINX] DO
2218   1   56:2    46            FOR BLK := BLOCK+START TO BLOCK+STOP-1 DO
2219   1   56:3    66              BEGIN        [ DO CONSECUTIVE READS OR WRITES ]
2220   1   56:4    66                IF INOUT = 'READ' THEN
2221   1   56:5    79                  UNITREAD(GUNIT,GBUF^[X],FBLKSIZE,BLK)
2222   1   56:4    92                ELSE
2223   1   56:5    94                  UNITWRITE(GUNIT,GBUF^[X],FBLKSIZE,BLK);
2224   1   56:4   107                IF IORESULT <> 0 THEN
2225   1   56:5   113                  BEGIN    [ TELL USER WHERE IN THE FILE AN ERROR OCCURRED ]
2226   1   56:6   113                    WRITE(INOUT,' ERROR, REL ',BLK-BLOCK,', ABS ',BLK);
2227   1   56:6   180                    EXIT(KRUNCH)
2228   1   56:5   184                  END;
2229   1   56:4   184                X := X+FBLKSIZE
2230   1   56:3   186              END
2231   1   56:0   192      END;
2232   1   56:0   216
2233   1   55:0     0    BEGIN
2234   1   55:1     0    WITH GDIR^[GINX] DO
2235   1   55:2     8      IF DFKIND <> XDSKFILE THEN
2236   1   55:3    18        BEGIN
2237   1   55:4    16          WRITE('MOVING ');
2238   1   55:4    35          IF DIRECTION = FOURWARD THEN
2239   1   55:5    40            WRITE('FORWARD')
2240   1   55:4    57          ELSE
2241   1   55:5    59            WRITE('BACK');
2242   1   55:4    73          WRITELN(' ',DTID);
2243   1   55:4    97          IF DTID = 'SYSTEM.PASCAL' THEN
2244   1   55:5   120            REBOOT := GVID = SYVID;          [ IS THIS BEING DONE ON ROOT DISK ? ]
2245   1   55:4   130          NBLOCKS := DLASTBLK-DFIRSTBLK;   [ NUMBER OF BLOCKS IN THE FILE       ]
2246   1   55:4   138          RELBLOCK := 0;                   [ RELATIVE BLOCK TO THE FILES        ]
2247   1   55:4   142          REPEAT
2248   1   55:5   142            CHUNKSIZE := NBLOCKS;          [ # OF BLOCKS LEFT TO MOVE           ]
```

```
2249  1   55:5   148            IF CHUNKSIZE > GBUFBLKS THEN
2250  1   55:6   155              CHUNKSIZE := GBUFBLKS;        [ THE BUFFER ISN'T BIG ENOUGH      ]
2251  1   55:5   159            NBLOCKS := NBLOCKS - CHUNKSIZE;[ BLOCKS LEFT TO MOVE             ]
2252  1   55:5   169            MOVEIT('READ',STARTING);      [ START READING THE FILE           ]
2253  1   55:5   180            MOVEIT('WRITE',STOPPING);     [ WRITE IT OUT SOMEWHERE ELSE      ]
2254  1   55:5   191            RELBLOCK := RELBLOCK+CHUNKSIZE;
2255  1   55:4   201          UNTIL NBLOCKS = 0;
2256  1   55:4   208          IF DIRECTION = REVERSE THEN
2257  1   55:5   213            OTHER := STOPPING - (DLASTBLK - DFIRSTBLK)   [ NEW DFIRSTBLK ]
2258  1   55:4   220          ELSE
2259  1   55:5   224            OTHER := STOPPING + (DLASTBLK - DFIRSTBLK);  [ NEW DLASTBLK ]
2260  1   55:4   233          STARTING := STOPPING;
2261  1   55:4   236          UPDATEDIR
2262  1   55:3   236        END
2263  1   55:0   238    END;
2264  1   55:0   254
2265  1   54:0     0  BEGIN [ KRUNCH ]
2266  1   54:1     0    CHECKFILE('CRUNCH','',1,BLKDEXP,FALSE,FALSE,[OKDIR]);
2267  1   54:1    24    GETBLOCKS('FROM END OF DISK, BLOCK','','STARTING AT BLOCK #',
2268  1   54:1    75                                          GDIR^ [0].DLASTBLK,SPLIT);
2269  1   54:1    84    WHICHFILE(SPLIT,FALSE);     [ WHICH FILES GO WHICH DIRECTION ? ]
2270  1   54:1    89    REBOOT := FALSE; [ WILL BE SET TO TRUE IF *SYSTEM.PASCAL IS MOVED ]
2271  1   54:1    92    SYSCOM^.MISCINFO.NOBREAK := TRUE; [ IGNORE ALL BREAK CHARS. DURING KRUNCH ]
2272  1   54:1   101    CLEARSCREEN;
2273  1   54:1   104    WRITELN;
2274  1   54:1   110    FOR GINX := 1 TO SPLIT-1 DO
2275  1   54:2   123      WITH GDIR^ [GINX] DO      [ MOVE THESE FILES TOWARDS THE FIRST BLOCK ]
2276  1   54:3   129        IF DFIRSTBLK > GDIR^ [GINX-1].DLASTBLK THEN
2277  1   54:4   141          KRUNCHIT(FOURWARD,DFIRSTBLK,DLASTBLK,GDIR^ [GINX-1].DLASTBLK);
2278  1   54:1   162    FIRSTBLK := GDIR^[0].DEOVBLK;
2279  1   54:1   169    FOR GINX := GDIR^[0].DNUMFILES DOWNTO SPLIT DO
2280  1   54:2   185      WITH GDIR^ [GINX] DO      [ MOVE THESE FILES TOWARDS THE LAST BLOCK  ]
2281  1   54:3   191        BEGIN
2282  1   54:4   191          IF DLASTBLK < FIRSTBLK THEN
2283  1   54:5   197            KRUNCHIT(REVERSE,DLASTBLK,DFIRSTBLK,FIRSTBLK);
2284  1   54:4   205          FIRSTBLK := DFIRSTBLK
2285  1   54:3   205        END;
2286  1   54:1   216    WRITELN(GVID,': CRUNCHED');
2287  1   54:1   251    IF REBOOT THEN
2288  1   54:2   254      BEGIN [ *SYSTEM.PASCAL WAS MOVED ]
2289  1   54:3   254        WRITELN('PLEASE RE-BOOT');
```

```
2290   1    54:3    284          REPEAT UNTIL FALSE
2291   1    54:2    284        END;
2292   1    54:1    287      SYSCOM^.MISCINFO.NOBREAK := FALSE
2293   1    54:0    294    END [ KRUNCH ];
2294   1    54:0    314
2295   1    54:0    314 [--------------------------- DRIVER ROUTINE --------------------------------]
2296   1    54:0    314
2297   1    54:0    314    [ THIS PROCEDURE IS THE MAIN CALLING SEGMENT. IT IS INVOKED IN AN INFINITE ]
2298   1    54:0    314    [ LOOP BY THE MAIN FILER SEGMENT.                                         ]
2299   1     2:0      1    PROCEDURE CALLPROC;
2300   1     2:0      1    VAR
2301   1     2:0      1      X,Y : INTEGER;
2302   1     2:0      3      OK : BOOLEAN;
2303   1     2:0      4
2304   1     2:0      4      [ CALLS OP-SYSTEM PROMPT ROUTINE ]
2305   1    57:0      1      PROCEDURE PROMPTEM(STR : STRING);
2306   1    57:0      0      BEGIN
2307   1    57:1      0        PL := STR;
2308   1    57:1     12        PROMPT;
2309   1    57:1     15        CH := GETCHAR(NOT OK);
2310   1    57:1     26        OK := CH IN [' ','?','B'..'E','G','K'..'N','P'..'T','V'..'X','Z']
2311   1    57:0     27      END;
2312   1    57:0     60
2313   1     2:0      0    BEGIN
2314   1     2:1      0      INITGLOBALS;
2315   1     2:1      2      INSTRING := '';
2316   1     2:1     10      OK := TRUE;
2317   1     2:1     13      REPEAT
2318   1     2:2     13        IF FAST THEN
2319   1     2:3     16          PROMPTEM(
2320   1     2:3     16 'FILER: G(ET, S(AVE, W(HAT, N(EW, L(DIR, R(EM, C(HNG, T(RANS, D(ATE, Q(UIT [II]')
2321   1     2:2     97        ELSE
2322   1     2:3    101          PROMPTEM('FILER: G, S, N, L, R, C, T, D, Q [II]');
2323   1     2:2    143        IF CH = '?' THEN
2324   1     2:3    148          IF FAST THEN
2325   1     2:4    151            PROMPTEM(
2326   1     2:4    151 'FILER: B(AD-BLKS, E(XT-DIR, K(RNCH, M(AKE, P(REFIX, V(OLS, X(AMINE, Z(ERO [II]')
2327   1     2:3    232          ELSE
2328   1     2:4    236            PROMPTEM('FILER: W, B, E, K, M, P, V, X, Z [II]')
2329   1     2:1    276      UNTIL OK;
2330   1     2:1    281      HOMECURSOR;
```

```
2331  1   2:1   284     CLEARLINE;
2332  1   2:1   287     IF CH IN [' ','B','D','Q','V','Z'] THEN
2333  1   2:2   306       CLEARSCREEN;
2334  1   2:1   309     FOR X := 1 TO 11 DO
2335  1   2:2   320     WITH UNITABLE[X] DO
2336  1   2:3   328       IF UVID <> '' THEN
2337  1   2:4   336         FOR Y := X+1 TO 12 DO
2338  1   2:5   349           IF UVID = UNITABLE[Y].UVID THEN
2339  1   2:6   360             BEGIN
2340  1   2:7   360               WRITEANDCLEAR;
2341  1   2:7   362               WRITE('WARNING UNITS ',X,' & ',Y,' HAVE THE SAME NAME');
2342  1   2:7   444               WRITEANDCLEAR
2343  1   2:6   444             END;
2344  1   2:1   460     CASE CH OF
2345  1   2:1   463       'L' : LISTDIR(FALSE);
2346  1   2:1   468       'E' : LISTDIR(TRUE);
2347  1   2:1   473       'G' : GETWORK;
2348  1   2:1   477       'N' : NEWWORK(TRUE);
2349  1   2:1   482       'C' : CHANGER;
2350  1   2:1   486       'R' : REMOVER;
2351  1   2:1   490       'T' : TRANSFER;
2352  1   2:1   494       'S' : IF SAVEWORK THEN TRANSFER;
2353  1   2:1   504       'P' : BEGIN
2354  1   2:3   504               CHECKFILE('PREFIX TITLES BY','',1,VOLEXP,FALSE,
2355  1   2:3   531                                       FALSE,[NOVOL,BADDIR,OKDIR,UNBLKDVOL]);
2356  1   2:3   538               DKVID := GVID;
2357  1   2:3   545               CLEARLINE;
2358  1   2:3   548               WRITE('PREFIX IS ',DKVID,':')
2359  1   2:2   586             END;
2360  1   2:1   588       'W' : WHATWORK;
2361  1   2:1   592       'M' : MAKEFILE;
2362  1   2:1   596       'V' : LISTVOLS;
2363  1   2:1   600       'B' : BADBLOCKS;
2364  1   2:1   604       'Z' : ZEROVOLUME;
2365  1   2:1   608       'X' : XBLOCKS;
2366  1   2:1   612       'K' : KRUNCH;
2367  1   2:1   616       'D' : DATESET;
2368  1   2:1   620       'Q' : EXIT(FILEHANDLER)
2369  1   2:1   624     END
2370  1   2:0   684   END;
2371  1   2:0   710
```

```
2372   1     2:0    710 {------------------------- MAIN SEGMENT ROUTINE ------------------------------}
2373   1     2:0    710
2374   1     1:0      0 BEGIN {FILEHANDLER}
2375   1     1:1      0   WITH USERINFO DO
2376   1     1:2     12     BEGIN      { INITIALIZE WORKFILES }
2377   1     1:3     12       TEXTSAVED := NOT GOTSYM OR (GOTSYM AND (SYMTID <> 'SYSTEM.WRK.TEXT'));
2378   1     1:3     46       CODESAVED := NOT GOTCODE OR (GOTCODE AND (CODETID <> 'SYSTEM.WRK.CODE'))
2379   1     1:2     77     END;
2380   1     1:1     80   FAST := (SYSCOM^.CRTINFO.WIDTH >= 80) AND (NOT SYSCOM^.MISCINFO.SLOWTERM);
2381   1     1:1     99   MARK(GBUF);  { SET UP TRANSFER BUFFER }
2382   1     1:1    103   GBUFBLKS := 0;
2383   1     1:1    106   REPEAT
2384   1     1:2    106     NEW(BLOCKPTR);
2385   1     1:2    113     GBUFBLKS := GBUFBLKS+1;
2386   1     1:2    118
2387   1     1:2    118   { LEAVE ROOM FOR FILER VARIABLES TO KEEP FROM STACK OVERFLOWING }
2388   1     1:1    118   UNTIL ((MEMAVAIL > 0) AND (MEMAVAIL < (SIZEOF(DIRECTORY)+SIZEOF(FIB)+1024)))
2389   1     1:1    137        OR (GBUFBLKS = 63); { BLOCK I/O LIMITATION }
2390   1     1:1    143
2391   1     1:1    143   { ABBREVIATIONS FOR THE MONTHS & FILE TYPES }
2392   1     1:1    143   MONTHSTR := '???JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC?????????';
2393   1     1:1    199   TYPESTR := '    BAD CODETEXTINFODATAGRAFFOTO';
2394   1     1:1    239
2395   1     1:1    239   REPEAT { CALL DRIVING ROUTINE IN AN INFINITE LOOP }
2396   1     1:2    239     CALLPROC;
2397   1     1:2    241
2398   1     1:2    241     { IN CASE WE ABORTED FROM TRANSFER AND LEFT A TEMP FILE }
2399   1     1:2    241     IF UNITABLE[DESTUNIT].UISBLKD THEN
2400   1     1:3    250       CLOSE(LFIB,PURGE)
2401   1     1:2    256     ELSE
2402   1     1:3    258       CLOSE(LFIB)
2403   1     1:1    264   UNTIL FALSE
2404   1     1:0    264 END;
2405   1     1:0    290
2406   0     1:0      0 BEGIN
2407   0     1:0      0 END.
```

```
 1   1    1:0    1 (*$L PRINTER:*)
 2   1    1:0    1 (*************************************************************************)
 3   1    1:0    1 (*                                                                       *)
 4   1    1:0    1 (*     SCREEN ORIENTED EDITOR                                            *)
 5   1    1:0    1 (*     ------ -------- ------                  WRITTEN: OCTOBER  11, 1978 *)
 6   1    1:0    1 (*                                             UPDATE : DECEMBER 10, 1978 *)
 7   1    1:0    1 (*                                                       ---------       *)
 8   1    1:0    1 (*     BY RICHARD S. KAUFMANN,                          /         \      *)
 9   1    1:0    1 (*        IIS                                           \ VERSION \      *)
10   1    1:0    1 (*        UNIVERSITY OF CALIFORNIA, SAN DIEGO           \   E.6F  \      *)
11   1    1:0    1 (*        LA JOLLA CA 92093                             \---------/      *)
12   1    1:0    1 (*                                                                       *)
13   1    1:0    1 (*     COPYRIGHT (C) 1978, BY THE REGENTS OF THE UNIVERSITY OF           *)
14   1    1:0    1 (*        CALIFORNIA AT SAN DIEGO                                        *)
15   1    1:0    1 (*                                                                       *)
16   1    1:0    1 (*************************************************************************)
16   1    1:0    1
17   1    1:0    1 (*$I HEAD *)
18   0    1:0    1 (*$U-*)
19   0    1:0    1 CONST
20   0    1:0    1   VIDLENG = 7;  (* NUMBER OF CHARACTERS IN A VOLUME ID *)
21   0    1:0    1   TIDLENG = 15; (* NUMBER OF CHARACTERS IN A TITLE ID *)
22   0    1:0    1
23   0    1:0    1 TYPE
24   0    1:0    1
25   0    1:0    1   DATEREC=PACKED RECORD
26   0    1:0    1             MONTH: 0..12;
27   0    1:0    1             DAY:   0..31;
28   0    1:0    1             YEAR:  0..100
29   0    1:0    1           END;
30   0    1:0    1
31   0    1:0    1   VID = STRING[VIDLENG];
32   0    1:0    1
33   0    1:0    1   TID = STRING[TIDLENG];
34   0    1:0    1
35   0    1:0    1   INFOREC = RECORD
36   0    1:0    1               TRASH1,TRASH2: INTEGER;
37   0    1:0    1               ERRSYM,ERRBLK,ERRNUM: INTEGER;     (* ERROR COM FOR EDIT *)
38   0    1:0    1               TRASH3: ARRAY [0..2] OF INTEGER;
39   0    1:0    1               GOTSYM,GOTCODE: BOOLEAN;
40   0    1:0    1               WORKVID,SYMVID,CODEVID: VID;       (* PERM&CUR WORKFILE VOLUMES *)
                                  WORKTID,SYMTID,CODETID: TID        (* PERM&CUR WORKFILE TITLES *)
```

```
41   0    1:0     1              END (*INFOREC*) ;
42   0    1:0     1
43   0    1:0     1   SYSCOMREC = RECORD
44   0    1:0     1                  JUNK: ARRAY [0..6] OF INTEGER;
45   0    1:0     1                  LASTMP: INTEGER;
46   0    1:0     1                  EXPANSION: ARRAY [0..20] OF INTEGER;
47   0    1:0     1                  MISCINFO: PACKED RECORD
48   0    1:0     1                              NOBREAK,STUPID,SLOWTERM,
49   0    1:0     1                              HASXYCRT,HASLCCRT,HAS8510A,HASCLOCK: BOOLEAN
50   0    1:0     1                            END;
51   0    1:0     1                  CRTTYPE: INTEGER;
52   0    1:0     1                  CRTCTRL: PACKED RECORD
53   0    1:0     1                              RLF,NDFS,ERASEEOL,ERASEEOS,HOME,ESCAPE: CHAR;
54   0    1:0     1                              BACKSPACE: CHAR;
55   0    1:0     1                              FILLCOUNT: 0..255;
56   0    1:0     1                              CLEARSCREEN,CLEARLINE: CHAR;
57   0    1:0     1                              PREFIXED: PACKED ARRAY [0..8] OF BOOLEAN
58   0    1:0     1                            END;
59   0    1:0     1                  CRTINFO: PACKED RECORD
60   0    1:0     1                              WIDTH,HEIGHT: INTEGER;
61   0    1:0     1                              RIGHT,LEFT,DOWN,UP: CHAR;
62   0    1:0     1                              BADCH,CHARDEL,STOP,BREAK,FLUSH,EOF: CHAR;
63   0    1:0     1                              ALTMODE,LINEDEL: CHAR;
64   0    1:0     1                              BACKSPACE,ETX,PREFIX: CHAR;
65   0    1:0     1                              PREFIXED: PACKED ARRAY [0..13] OF BOOLEAN;
66   0    1:0     1                            END
67   0    1:0     1                END (*SYSCOM*);
68   0    1:0     1
69   0    1:0     1 VAR (* I.4 GLOBALS AS OF 30-JAN-78 *)
70   0    1:0     1   SYSCOM: ^SYSCOMREC;
71   0    1:0     2   TRASHY: ARRAY [0..5] OF INTEGER;
72   0    1:0     8   USERINFO: INFOREC;
73   0    1:0    54   TRASHYY: ARRAY [0..4] OF INTEGER;
74   0    1:0    59   SYVID,DKVID: VID;
75   0    1:0    67   THEDATE: DATEREC;
76   0    1:0    68
77   0    1:0    68
78   1    1:0     1 SEGMENT PROCEDURE EDITOR(XXX,YYY: INTEGER);
79   1    1:0     3 CONST
80   1    1:0     3   (* UNLESS OTHERWISE NOTED ALL CONSTANTS ARE UPPER BOUNDS
81   1    1:0     3      FROM ZERO.                                          *)
```

```
 82   1     1:D      3
 83   1     1:D      3   MAXBUFSIZE=32767;
 84   1     1:D      3   MAXSW=84; (* MAXIMUM ALLOWABLE SCREENWIDTH *)
 85   1     1:D      3   MAXSTRING=127;
 86   1     1:D      3   MAXCHAR=1023;  (* THE MAXIMUM NUMBER OF CHARACTERS ON A LINE IN THE EBUF *)
 87   1     1:D      3   TIDLENG=15;  (* FROM SYSCOM *)
 88   1     1:D      3   CHARINBUF=2048;  (* FOR FINAL VERSION. NOT USED. *)
 89   1     1:D      3   MAXOFFSET=1023; (* MAXIMUM OFFSET IN A PAGE *)
 90   1     1:D      3   MAXPAGE=255; (* RIDICULOUS UPPER BOUND! *)
 91   1     1:D      3
 92   1     1:D      3   (* THE FOLLOWING ASCII CHARACTERS ARE HARD-WIRED IN *)
 93   1     1:D      3   HT=9; LF=10; EOL=13; DLE=16; SP=32;
 94   1     1:D      3   DC1=17; BELL=7; RUBOUT=127; CR=13;
 95   1     1:D      3
 96   1     1:D      3
 97   1     1:D      3 TYPE
 98   1     1:D      3   PTRTYPE=0..MAXBUFSIZE;
 99   1     1:D      3   BUFRTYPE=PACKED ARRAY [0..0] OF CHAR;
100   1     1:D      3   BLOCKTYPE=PACKED ARRAY [0..511] OF CHAR;
101   1     1:D      3   ERRORTYPE=(FATAL,NONFATAL);
102   1     1:D      3   OFFSET=0..MAXOFFSET;
103   1     1:D      3   PAGE=0..MAXPAGE;
104   1     1:D      3   NAME=PACKED ARRAY [0..7] OF CHAR;
105   1     1:D      3   PTYPE=PACKED ARRAY [0..MAXSTRING] OF CHAR;
106   1     1:D      3   COMMANDS=(ILLEGAL,ADJUSTC,COPYC,DELETEC,FINDC,INSERTC,JUMPC,LISTC,MACRODEFC,
107   1     1:D      3             PARAC,QUITC,REPLACEC,SETC,VERIFYC,XECUTEC,ZAPC,REVERSEC,FORWARDC,UP,
108   1     1:D      3             DOWN,LEFT,RIGHT,TAB,DIGIT,DUMPC,ADVANCE,SPACE,EQUALC,SLASHC);
109   1     1:D      3   CTYPE=(FS,GOHOME,ETOEOL,ETOEOS,US);
110   1     1:D      3   SCREENCOMMAND=(WHOME,ERASEEOS,ERASEEOL,CLEARLNE,CLEARSCN,UPCURSOR,
111   1     1:D      3                  DOWNCURSOR,LEFTCURSOR,RIGHTCURSOR);
112   1     1:D      3   KEYCOMMAND=   (BACKSPACEKEY,DC1KEY,EOFKEY,ETXKEY,ESCAPEKEY,DELKEY,UPKEY,
113   1     1:D      3                  DOWNKEY,LEFTKEY,RIGHTKEY,NOTLEGAL);
114   1     1:D      3
115   1     1:D      3   HEADER=     (* PAGE ZERO LAYOUT CHANGED 22-JUN-78 *)
116   1     1:D      3     RECORD CASE BOOLEAN OF
117   1     1:D      3       TRUE: (BUF: PACKED ARRAY[0..MAXOFFSET] OF CHAR);
118   1     1:D      3       FALSE:(DEFINED:    BOOLEAN; (* NEW FILE NULLS => FALSE *)
119   1     1:D      3              COUNT:      INTEGER;  (* THE COUNT OF VALID MARKERS *)
120   1     1:D      3              NAME:       ARRAY [0..9] OF PACKED ARRAY [0..7] OF CHAR;
121   1     1:D      3              PAGEN:      PACKED ARRAY [0..9] OF PAGE;
122   1     1:D      3              POFFSET:    PACKED ARRAY [0..9] OF OFFSET;
```

```
123   1    1:D    3              AUTOINDENT: BOOLEAN; (* ENVIRONMENT STUFF FOLLOWS *)
124   1    1:D    3              FILLING:    BOOLEAN;
125   1    1:D    3              TOKDEF:     BOOLEAN;
126   1    1:D    3              LMARGIN:    0..MAXSW;
127   1    1:D    3              RMARGIN:    0..MAXSW;
128   1    1:D    3              PARAMARGIN: 0..MAXSW;
129   1    1:D    3              RUNOFFCH:   CHAR;
130   1    1:D    3              CREATED:    DATEREC;
131   1    1:D    3              LASTUSED:   DATEREC;
132   1    1:D    3              FILLER:     PACKED ARRAY [0..891] OF CHAR)
133   1    1:D    3     END;
134   1    1:D    3
135   1    1:D    3
136   1    1:D    3
137   1    1:D    3 VAR
138   1    1:D    3   CURSOR: 0..MAXBUFSIZE;
139   1    1:D    4   BUFCOUNT: 0..MAXBUFSIZE;        (* NUMBER OF VALID CHARACTERS IN THE EBUF *)
140   1    1:D    5   STUFFSTART: 0..MAXBUFSIZE;      (* GETLEADING              *)
141   1    1:D    6   LINESTART: 0..MAXBUFSIZE;      (*             SETS        *)
142   1    1:D    7   BYTES,BLANKS: INTEGER;          (*                  THESE *)
143   1    1:D    9   CH: CHAR;
144   1    1:D   10   DIRECTION: CHAR;                (* '>' OR '<' *)
145   1    1:D   11   REPEATFACTOR: INTEGER;
146   1    1:D   12   BUFSIZE: INTEGER;
147   1    1:D   13   SCREENWIDTH: INTEGER;           (* MOVED TO VAR 26-JAN *)
148   1    1:D   14   SCREENHEIGHT: INTEGER;          (*   "    "  "    "    *)
149   1    1:D   15   COMMAND: COMMANDS;
150   1    1:D   16   LASTPAT: 0..MAXBUFSIZE;
151   1    1:D   17   EBUF: ^BUFRTYPE;
152   1    1:D   18   FILLIT: STRING[11];
153   1    1:D   24   KIND: ARRAY [CHAR] OF INTEGER; (* FOR TOKEN FIND *)
154   1    1:D   80   LINE1PTR: 0..MAXBUFSIZE;
155   1    1:D   81   MIDDLE: INTEGER;               (* MIDDLE LINE ON THE SCREEN *)
156   1    1:D   82   NEEDPROMPT: BOOLEAN;
157   1    1:D   83   ETX,BS,DEL,ESC,BSPCE: INTEGER;  (* MOVED FROM CONST 30-JAN-78 BSPCE: 11/2/78*)
158   1    1:D   88   ADJUSTPROMPT,INSERTPROMPT,DELETEPROMPT,COMPROMPT: STRING;
159   1    1:D   52   [PROMPTLINE  11/2/78 M. BERNARD]
160   1    1:D   52   TRASH: INTEGER;                 (* TOTALLY WITHOUT REDEEMING SOCIAL VALUE *)
161   1    1:D   53   TARGET: PTYPE;
162   1    1:D   17   SUBSTRING: PTYPE;
163   1    1:D   81   SLENGTH,TLENGTH: INTEGER;       (* LENGTH OF TARGET AND SUBSTRING *)
```

```
164   1    1:D     83   SDEFINED,TDEFINED: BOOLEAN;     (* WHETHER THE STRINGS ARE VALID *)
165   1    1:D     85   COPYLENGTH,COPYSTART: PTRTYPE; (* FOR COPYC *)
166   1    1:D     87   COPYLINE,COPYOK: BOOLEAN;       (*     "       *)
167   1    1:D     89   INFINITY: BOOLEAN;              (* FOR SLASHC *)
168   1    1:D     90   THEFILE: FILE;
169   1    1:D     30   TRANSLATE: ARRAY [CHAR] OF COMMANDS;
170   1    1:D     86   PAGEZERO: HEADER;
171   1    1:D   1 98   MSG: STRING;
172   1    1:D   1 39   PROMPTLINE: STRING;
173   1    1:D   1 80   BLANKAREA: ARRAY [0..MAXSW] OF CHAR;
174   1    1:D   1 65   SAVETOP: STRING;                (* DUMB TERMINAL PATCH - FOR BLANKCRT(1) *)
175   1    1:D   1 06   SCREEN: PACKED RECORD (* SCREEN CONTROL RECORD *)
176   1    1:D   1 06             PREFIX: CHAR;
177   1    1:D   1 06             HEIGHT,WIDTH: 0..255;
178   1    1:D   1 06             CANUPSCROLL,CANDOWNSCROLL,SLOW: BOOLEAN;
179   1    1:D   1 06             HASPREFIX: PACKED ARRAY [SCREENCOMMAND] OF BOOLEAN;
180   1    1:D   1 06             CH:        PACKED ARRAY [SCREENCOMMAND] OF CHAR
181   1    1:D   1 06           END;
182   1    1:D   1 14   KEYBRD: PACKED RECORD (* KEYBOARD CONTROL RECORD *)
183   1    1:D   1 14             PREFIX: CHAR;
184   1    1:D   1 14             HASPREFIX: PACKED ARRAY [KEYCOMMAND] OF BOOLEAN;
185   1    1:D   1 14             CH:        PACKED ARRAY [KEYCOMMAND] OF CHAR
186   1    1:D   1 14           END;
187   1    1:D   1 22
188   3    1:5     0 SEGMENT PROCEDURE NUM2; BEGIN END;       SEGMENT PROCEDURE NUM3; BEGIN END;
189   5    1:0     0 SEGMENT PROCEDURE NUM4; BEGIN END;       SEGMENT PROCEDURE NUM5; BEGIN END;
190   7    1:0     0 SEGMENT PROCEDURE NUM6; BEGIN END;       SEGMENT PROCEDURE NUM7; BEGIN END;
191   9    1:0     0 SEGMENT PROCEDURE NUM8; BEGIN END;       SEGMENT PROCEDURE NUM9; BEGIN END;
192   9    1:0    12
193   9    1:0    12
194   9    1:0    12
195   9    1:0    12 (*$I HEAD *)
196   9    1:0    12
197   9    1:0    12 (* FORWARD DECLARED PROCEDURES.. ALL PROCEDURES ARE IN MISC AND UTIL *)
198   9    1:0    12
199   1    2:D     1 PROCEDURE ERROR(S:STRING;HOWBAD:ERRORTYPE); FORWARD;
200   1    3:D     1 PROCEDURE ERASETOEOL(X,LINE:INTEGER); FORWARD;
201   1    4:D     3 FUNCTION  GETCH:CHAR; FORWARD;
202   1    5:D     1 PROCEDURE CLEARSCREEN; FORWARD;
203   1    6:D     1 PROCEDURE ERASEOS(X,LINE:INTEGER); FORWARD;
204   1    7:D     1 PROCEDURE CLEARLINE(Y:INTEGER); FORWARD;
```

```
205   1    8:D     3 FUNCTION  MAPTOCOMMAND(CH:CHAR): COMMANDS; FORWARD;
206   1    9:D     3 FUNCTION  UCLC(CH:CHAR): CHAR; FORWARD;
207   1   10:D     1 PROCEDURE PROMPT; FORWARD;
208   1   11:D     1 PROCEDURE REDISPLAY; FORWARD;
209   1   12:D     3 FUNCTION  MIN(A,B:INTEGER): INTEGER; FORWARD;
210   1   13:D     3 FUNCTION  MAX(A,B:INTEGER): INTEGER; FORWARD;
211   1   14:D     3 FUNCTION  SCREENHAS(WHAT: SCREENCOMMAND): BOOLEAN; FORWARD;
212   1   15:D     3 FUNCTION  HASKEY(WHAT: KEYCOMMAND): BOOLEAN; FORWARD;
213   1   16:D     1 PROCEDURE CONTROL(WHAT: SCREENCOMMAND); FORWARD;
214   1   17:D     1 PROCEDURE PUTMSG; FORWARD;
215   1   18:D     1 PROCEDURE HOME; FORWARD;
216   1   19:D     1 PROCEDURE ERRWAIT; FORWARD;
217   1   20:D     1 PROCEDURE BLANKCRT(Y: INTEGER); FORWARD;
218   1   21:D     3 FUNCTION  LEADBLANKS(PTR:PTRTYPE;VAR BYTES: INTEGER): INTEGER; FORWARD;
219   1   22:D     1 PROCEDURE CENTERCURSOR(VAR LINE: INTEGER; LINESUP: INTEGER; NEWSCREEN:BOOLEAN);
220   1   22:D     4   FORWARD;
221   1   23:D     1 PROCEDURE FINDXY(VAR INDENT,LINE: INTEGER); FORWARD;
222   1   24:D     1 PROCEDURE SHOWCURSOR; FORWARD;
223   1   25:D     3 FUNCTION  GETNUM: INTEGER; FORWARD;
224   1   26:D     1 PROCEDURE GETLEADING; FORWARD;
225   1   27:D     3 FUNCTION  OKTODEL(CURSOR,ANCHOR:PTRTYPE):BOOLEAN; FORWARD;
226   1   28:D     1 PROCEDURE LINEOUT(VAR PTR:PTRTYPE; BYTES,BLANKS,LINE: INTEGER); FORWARD;
227   1   29:D     1 PROCEDURE UPSCREEN(FIRSTLINE,WHOLESCREEN:BOOLEAN; LINE: INTEGER); FORWARD;
228   1   30:D     1 PROCEDURE READJUST(CURSOR: PTRTYPE; DELTA: INTEGER); FORWARD;
229   1   31:D     1 PROCEDURE THEFIXER(PARAPTR: PTRTYPE;RFAC: INTEGER;WHOLE: BOOLEAN); FORWARD;
230   1   32:D     1 PROCEDURE GETNAME(MSG:STRING; VAR M:NAME); FORWARD;
231   1   32:D    44
231   1   32:D    44 (*$I INIT      *)
232  10    1:D     1 SEGMENT PROCEDURE INITIALIZE;
233  10    1:D     1 LABEL 1;
234  10    1:D     1 VAR
235  10    1:D     1   BLOCK: ^BLOCKTYPE;
236  10    1:D     2   ONEWD: ^INTEGER;
237  10    1:D     3   DONE,OVFLW: BOOLEAN;
238  10    1:D     5   CH: CHAR;
239  10    1:D     6   I,QUIT,GAP,BLKS,PAGE,NOTNULS: INTEGER;
240  10    1:D    12   FILENAME: STRING;
241  10    1:D    53   BUFFER: PACKED ARRAY [0..1023] OF CHAR;
242  10    1:D    65
243  10    2:D     1 PROCEDURE MAP(CH:CHAR; C:COMMANDS);
244  10    2:0     0 BEGIN
```

```
245  10    2:1     0    TRANSLATE[CH]:=C;
246  10    2:1     8    IF CH IN ['A'..'Z'] THEN TRANSLATE[CHR(32+ORD(CH))]:=C; (* LC TOO *)
247  10    2:0    38  END;
248  10    2:0    50
249  10    3:D     1  PROCEDURE DEFPROMPTS; (* DEFINES VARIABLE PROMPTLINES MAB 11/2/78*)
250  10    3:0     0  BEGIN
251  10    3:1     0    COMPROMPT:=
252  10    3:1     3  ' EDIT: A(DJST C(PY D(LETE F(IND I(NSRT J(MP R(PLACE Q(UIT X(CHNG Z(AP  [E.6F]';
253  10    3:1    85    INSERTPROMPT:=
254  10    3:1    88    ' INSERT: TEXT [<BS> A CHAR,<DEL> A LINE]  [<ETX> ACCEPTS, <ESC> ESCAPES]';
255  10    3:1    65    DELETEPROMPT:=
256  10    3:1    68    ' DELETE: < > <MOVING COMMMANDS> [<ETX> TO DELETE, <ESC> TO ABORT]';
257  10    3:1    38    ADJUSTPROMPT:=
258  10    3:1    41    ' ADJUST: L(JUST R(JUST C(ENTER <LEFT,RIGHT,UP,DOWN-ARROWS> [<ETX> TO LEAVE]';
259  10    3:1    21    IF (SCREENWIDTH+1)<LENGTH(COMPROMPT) THEN
260  10    3:2    32    BEGIN
261  10    3:3    32      INSERTPROMPT:=' INSRT: [<BS>,<DEL>] [<ETX> ACCEPTS, <ESC> ABORTS]';
262  10    3:3    90      DELETEPROMPT:=' DELETE: <VECTOR KEYS> [<ETX> DELETES, <ESC> ABORTS]';
263  10    3:3    50      ADJUSTPROMPT:=' ADJST: L(FT R(GHT C(NTR <VECTOR KEYS> <ETX> TO LEAVE';
264  10    3:3    11      COMPROMPT:=' EDIT: A, C, D, F, I, J, R, Q, X, Z  [E.6F]';
265  10    3:2    62    END;
266  10    3:0    62  END;
267  10    3:0    76
268  10    4:D     1  PROCEDURE READFILE;
269  10    4:0     0  BEGIN
270  10    4:1     0    CLEARSCREEN;           (* DUMB TERMINAL PATCH *)
271  10    4:1     3    WRITELN('>EDIT:');
272  10    4:1    25    WRITE('READING');
273  10    4:1    42    IF BLOCKREAD(THEFILE,PAGEZERO,2)<>2 THEN ERROR('READING FILE',FATAL);
274  10    4:1    83    WRITE('.');
275  10    4:1    91    PAGE:=1;
276  10    4:1    95    DONE:=FALSE; OVFLW:=FALSE;
277  10    4:1    03    WITH USERINFO DO
278  10    4:2    03      WHILE NOT (DONE OR OVFLW) DO
279  10    4:3    13        BEGIN
280  10    4:4    13          DONE:= BLOCKREAD(THEFILE,BUFFER,2)=0;
281  10    4:4    36          IF NOT DONE THEN
282  10    4:5    42            BEGIN
283  10    4:6    42              WRITE('.');
284  10    4:6    50              NOTNULS:=SCAN(-1024,<>CHR(0),BUFFER[1023])+1024;
285  10    4:6    72              OVFLW:=NOTNULS+BUFCOUNT>=BUFSIZE-10;
```

```
286  10    4:6    84              IF OVFLW THEN NOTNULS:=0;
287  10    4:6    93              MOVELEFT(BUFFER[0],EBUF^[BUFCOUNT],NOTNULS);
288  10    4:6    05              IF PAGE+PAGE=ERRBLK THEN CURSOR:=BUFCOUNT+ERRSYM; (* ERRBLK>0 ONLY *)
289  10    4:6    25              BUFCOUNT:=BUFCOUNT+NOTNULS;
290  10    4:6    32              PAGE:=PAGE+1;
291  10    4:5    40            END;
292  10    4:3    40        END;
293  10    4:1    42    IF IORESULT<>0 THEN ERROR('DISK ERROR',NONFATAL) ELSE
294  10    4:2    67      IF NOT DONE THEN ERROR('BUFFER OVERFLOW.',NONFATAL);
295  10    4:0    96 END;
296  10    4:0    12
297  10    5:D     1 PROCEDURE LOADFROMSYSCOM;
298  10    5:D     1 (* A RATHER PERVERTED PROCEDURE THAT TAKES THE SYSCOM^.CRTCNTRL RECORD
299  10    5:D     1    AND LOADS IT INTO THE SCREEN CONTROL RECORD AND THE SYSCOM^.CRTINFO
300  10    5:D     1    RECORD AND LOADS IT INTO THE KEYBOARD CONTROL RECORD *)
301  10    5:0     0 BEGIN
302  10    5:0     0
303  10    5:1     0   WITH SYSCOM^ DO
304  10    5:2     5     BEGIN
305  10    5:2     5
306  10    5:2     5       (* MISCELLANEOUS STUFF *)
307  10    5:2     5
308  10    5:3     5       WITH SCREEN DO
309  10    5:4     5         BEGIN
310  10    5:5     5           PREFIX:=CRTCTRL.ESCAPE;
311  10    5:5    14           HEIGHT:=CRTINFO.HEIGHT-1;
312  10    5:5    25           WIDTH:=CRTINFO.WIDTH-1;
313  10    5:5    36           CANUPSCROLL:=TRUE; CANDOWNSCROLL:=FALSE;
314  10    5:4    50         END;
315  10    5:4    50
316  10    5:3    50       KEYBRD.PREFIX:=CRTINFO.PREFIX;
317  10    5:3    59
318  10    5:3    59       (* THE SCREEN ... *)
319  10    5:3    59
320  10    5:3    59       SCREEN.CH[WHOME]:=CRTCTRL.HOME;
321  10    5:3    70       SCREEN.HASPREFIX[WHOME]:=CRTCTRL.PREFIXED[4];
322  10    5:3    86
323  10    5:3    86       SCREEN.CH[ERASEEOS]:=CRTCTRL.ERASEEOS;
324  10    5:3    97       SCREEN.HASPREFIX[ERASEEOS]:=CRTCTRL.PREFIXED[3];
325  10    5:3    13
326  10    5:3    13       SCREEN.CH[ERASEEOL]:=CRTCTRL.ERASEEOL;
```

```
327  10    5:3    24        SCREEN.HASPREFIX[ERASEEOL]:=CRTCTRL.PREFIXED[2];
328  10    5:3    40
329  10    5:3    40        SCREEN.CH[CLEARLNE]:=CRTCTRL.CLEARLINE;
330  10    5:3    51        SCREEN.HASPREFIX[CLEARLNE]:=CRTCTRL.PREFIXED[7];
331  10    5:3    67
332  10    5:3    67        SCREEN.CH[CLEARSCN]:=CRTCTRL.CLEARSCREEN;
333  10    5:3    78        SCREEN.HASPREFIX[CLEARSCN]:=CRTCTRL.PREFIXED[6];
334  10    5:3    94
335  10    5:3    94        SCREEN.CH[UPCURSOR]:=CRTCTRL.RLF;
336  10    5:3    05        SCREEN.HASPREFIX[UPCURSOR]:=CRTCTRL.PREFIXED[0];
337  10    5:3    21
338  10    5:3    21        SCREEN.CH[DOWNCURSOR]:=CHR(LF);
339  10    5:3    27        SCREEN.HASPREFIX[DOWNCURSOR]:=FALSE;
340  10    5:3    36
341  10    5:3    36        SCREEN.CH[LEFTCURSOR]:=CRTCTRL.BACKSPACE;
342  10    5:3    47        SCREEN.HASPREFIX[LEFTCURSOR]:=CRTCTRL.PREFIXED[1];
343  10    5:3    63
344  10    5:3    63        SCREEN.CH[RIGHTCURSOR]:=CRTCTRL.NDFS;
345  10    5:3    74        SCREEN.HASPREFIX[RIGHTCURSOR]:=CRTCTRL.PREFIXED[8];
346  10    5:3    90
347  10    5:3    90        (* ... AND THE KEYBOARD *)
348  10    5:3    90
349  10    5:3    90        KEYBRD.CH[BACKSPACEKEY]:=CRTINFO.BACKSPACE;
350  10    5:3    98        KEYBRD.HASPREFIX[BACKSPACEKEY]:=CRTINFO.PREFIXED[12];
351  10    5:3    14
352  10    5:3    14        KEYBRD.CH[DC1KEY]:=CHR(DC1); (* NOT IN RECORD *)
353  10    5:3    20        KEYBRD.HASPREFIX[DC1KEY]:=FALSE;
354  10    5:3    29
355  10    5:3    29        KEYBRD.CH[EOFKEY]:=CRTINFO.EOF;
356  10    5:3    40        KEYBRD.HASPREFIX[EOFKEY]:=CRTINFO.PREFIXED[9];
357  10    5:3    56
358  10    5:3    56        KEYBRD.CH[ETXKEY]:=CRTINFO.ETX;
359  10    5:3    67        KEYBRD.HASPREFIX[ETXKEY]:=CRTINFO.PREFIXED[13];
360  10    5:3    83
361  10    5:3    83        KEYBRD.CH[ESCAPEKEY]:=CRTINFO.ALTMODE;
362  10    5:3    94        KEYBRD.HASPREFIX[ESCAPEKEY]:=CRTINFO.PREFIXED[10];
363  10    5:3    10
364  10    5:3    10        KEYBRD.CH[DELKEY]:=CRTINFO.LINEDEL;
365  10    5:3    21        KEYBRD.HASPREFIX[DELKEY]:=CRTINFO.PREFIXED[11];
366  10    5:3    37
367  10    5:3    37        KEYBRD.CH[UPKEY]:=CRTINFO.UP;
```

```
368  10    5:3    48        KEYBRD.HASPREFIX[UPKEY]:=CRTINFO.PREFIXED[3];
369  10    5:3    64
370  10    5:3    64        KEYBRD.CH[DOWNKEY]:=CRTINFO.DOWN;
371  10    5:3    75        KEYBRD.HASPREFIX[DOWNKEY]:=CRTINFO.PREFIXED[2];
372  10    5:3    91
373  10    5:3    91        KEYBRD.CH[LEFTKEY]:=CRTINFO.LEFT;
374  10    5:3    02        KEYBRD.HASPREFIX[LEFTKEY]:=CRTINFO.PREFIXED[1];
375  10    5:3    18
376  10    5:3    18        KEYBRD.CH[RIGHTKEY]:=CRTINFO.RIGHT;
377  10    5:3    29        KEYBRD.HASPREFIX[RIGHTKEY]:=CRTINFO.PREFIXED[0];
378  10    5:3    45
379  10    5:3    45        BSPCE:=ORD(CRTINFO.BACKSPACE);  [WENT SOFT 11/2/78 M. BERNARD]
380  10    5:3    51
381  10    5:3    51        [NOW TEST TO SEE THAT THE ESSENTIAL KEYS HAVE BEEN GIVEN A
382  10    5:3    51         VALUE OTHER THAN NULL.  IF NOT THEN ASSIGN THEM A DEFAULT
383  10    5:3    51         VALUE.  HOPEFULLY, THIS WILL END UP AN INTERP CHANGE--M. BERNARD]
384  10    5:3    51
385  10    5:3    51         IF BSPCE=0 THEN BSPCE:=8;
386  10    5:3    62         IF KEYBRD.CH[ETXKEY]=CHR(0) THEN KEYBRD.CH[ETXKEY]:=CHR(3);
387  10    5:3    77
388  10    5:3    77
389  10    5:2    77     END;
390  10    5:0    77 END;
391  10    5:0    90
392  10    6:D     1 PROCEDURE MAPSPECIAL(K:KEYCOMMANDS;C:COMMANDS);
393  10    6:0     0 BEGIN
394  10    6:1     0   IF NOT KEYBRD.HASPREFIX[K] THEN MAP(KEYBRD.CH[K],C);
395  10    6:0    19 END;
396  10    6:0    32
397  10    1:0     0 BEGIN
398  10    1:1     0   WITH PAGEZERO DO
399  10    1:2     0     BEGIN
400  10    1:2     0
401  10    1:2     0       (* LOAD SCREEN AND KEYBOARD CONTROL RECORDS FROM SYSCOM *)
402  10    1:2     0
403  10    1:3     0       LOADFROMSYSCOM;
404  10    1:3     2
405  10    1:3     2
406  10    1:3     2       (* INIT THE TRANSLATE TABLE *)
407  10    1:3     2
408  10    1:3     2       FILLCHAR(TRANSLATE,SIZEOF(TRANSLATE),ILLEGAL);
```

```
409  10    1:3    12      MAP('A',ADJUSTC);   MAP('C',COPYC);     MAP('D',DELETEC);
410  10    1:3    24      MAP('F',FINDC);     MAP('I',INSERTC);   MAP('J',JUMPC);
411  10    1:3    36      MAP('L',LISTC);     MAP('M',MACRODEFC); MAP('P',PARAC);
412  10    1:3    48      MAP('Q',QUITC);     MAP('R',REPLACEC);  MAP('S',SETC);
413  10    1:3    60      MAP('V',VERIFYC);   MAP('X',XECUTEC);   MAP('Z',ZAPC);
414  10    1:3    72      MAP(',',REVERSEC);  MAP('>',FORWARDC);  MAP('.',FORWARDC);
415  10    1:3    84      MAP('+',FORWARDC);  MAP('-',REVERSEC);  MAP('?',DUMPC);
416  10    1:3    96      MAP('/',SLASHC);    MAP('=',EQUALC);    MAP('<',REVERSEC);
417  10    1:3    08
418  10    1:3    08
419  10    1:3    08      (* ARROWS *)
420  10    1:3    08
421  10    1:3    08      (* NEXTCOMMAND AND GETNUM HANDLE VT-52 STYLE VECTOR KEYS *)
422  10    1:3    08      WITH KEYBRD DO
423  10    1:4    08        BEGIN
424  10    1:5    08          MAPSPECIAL(UPKEY,UP); MAPSPECIAL(DOWNKEY,DOWN);
425  10    1:5    16          MAPSPECIAL(LEFTKEY,LEFT); MAPSPECIAL(RIGHTKEY,RIGHT);
426  10    1:4    24        END;
427  10    1:3    24      MAP(CHR(EOL),ADVANCE); (* CR IS ADVANCE *)
428  10    1:3    28      MAP(CHR(HT),TAB);
429  10    1:3    32      MAP(CHR(SP),SPACE);
430  10    1:3    36
431  10    1:3    36
432  10    1:3    36      (* DIGITS *)
433  10    1:3    36
434  10    1:3    36      FOR CH:='0' TO '9' DO MAP(CH,DIGIT);
435  10    1:3    61
436  10    1:3    61
437  10    1:3    61      (* VARIABLE BUFFER SIZING... ADDED 17-JAN-78 *)
438  10    1:3    61
439  10    1:3    61      QUIT:=10512+          (* SIZEOF(EDITCORE)-SIZEOF(INITIALIZE) *)
440  10    1:3    64            512;          (* SLOP! *)
441  10    1:3    70      MARK(EBUF);
442  10    1:3    74      BLKS:=0;
443  10    1:3    77      REPEAT
444  10    1:4    77        NEW(BLOCK);
445  10    1:4    84        BLKS:=BLKS+1;
446  10    1:4    89        GAP:=MEMAVAIL+MEMAVAIL
447  10    1:3    91      UNTIL ((GAP>0) AND (GAP<QUIT)) OR (BLKS=63);
448  10    1:3    09      BUFSIZE:=BLKS*512-1;
449  10    1:3    18      NEW(ONEWD); ONEWD^:=0;  (* SENTINEL FOR END OF BUFFER - FOR M(UNCH *)
```

```
450  10     1:3     26
451  10     1:3     26
452  10     1:3     26
453  10     1:3     26
454  10     1:3     26
455  10     1:3     26
456  10     1:3     33
457  10     1:4     45
458  10     1:3     56
459  10     1:4     59
460  10     1:3     64
461  10     1:3     73
462  10     1:3     78
463  10     1:3     81
464  10     1:3     84
465  10     1:3     87
466  10     1:3     09
467  10     1:4     14
468  10     1:5     14
469  10     1:5     60
470  10     1:4     84
471  10     1:3     87
472  10     1:4     89
473  10     1:5     89
474  10     1:5     92
475  10     1:5     67
476  10     1:6     67
477  10     1:6     83
478  10     1:6     95
479  10     1:6     10
480  10     1:7     18
481  10     1:8     18
482  10     1:7     30
483  10     1:6     30
484  10     1:6     49
485  10     1:6     49
486  10     1:6     86
487  10     1:6     08
488  10     1:7     27
489  10     1:6     62
490  10     1:7     73
```

```
(* OPEN THE WORKFILE *)

(*INIT FILLIT FOR WRITING NULLS IN FRONT OF ALL CONTROL CH'S*)
FILLCHAR(FILLIT,SIZEOF(FILLIT),0);
IF SYSCOM^.CRTCTRL.FILLCOUNT<=11 THEN
  FILLIT[0]:=CHR(SYSCOM^.CRTCTRL.FILLCOUNT)
ELSE
  FILLIT[0]:=CHR(11);
FILLCHAR(EBUF^,BUFSIZE+1,CHR(0));
EBUF^[0]:=CHR(EOL);
BUFCOUNT:=1;
CURSOR:=1;
CLEARSCREEN;
WRITELN('>EDIT:');
IF USERINFO.GOTSYM THEN
  BEGIN
    RESET(THEFILE,CONCAT(USERINFO.SYMVID,':',USERINFO.SYMTID));
    IF IORESULT<>0 THEN ERROR('WORKFILE LOST.',FATAL)
  END
ELSE
  BEGIN
    MSG:=
'NO WORKFILE IS PRESENT. FILE? ( <RET> FOR NO FILE <ESC-RET> TO EXIT ) ';
    REPEAT
      WRITELN(MSG);
      WRITE(': ');
      READLN(INPUT,FILENAME);
      IF LENGTH(FILENAME)=0 THEN
        BEGIN
          FILLCHAR(PAGEZERO,SIZEOF(PAGEZERO),CHR(0)); GOTO 1;
        END;
      IF FILENAME[1]=SYSCOM^.CRTINFO.ALTMODE THEN EXIT(EDITOR);
      [ TO ESCAPE IF ENTERED BY ACCIDENT. MAB 12/8/78 ]
      FOR I:=1 TO LENGTH(FILENAME) DO FILENAME[I]:=UCLC(FILENAME[I]);
      IF ((POS('.TEXT',FILENAME)<>LENGTH(FILENAME)-4) OR
         (LENGTH(FILENAME)<=4)) AND (FILENAME[LENGTH(FILENAME)]<>'.') THEN
        FILENAME:=CONCAT(FILENAME,'.TEXT');
      IF FILENAME[LENGTH(FILENAME)]='.' THEN
        DELETE(FILENAME,LENGTH(FILENAME),1);
```

```
491  10    1:6    83          OPENOLD(THEFILE,FILENAME);
492  10    1:6    93          MSG:='NOT PRESENT. FILE? ';
493  10    1:5    20        UNTIL IORESULT=0;
494  10    1:4    26      END;
495  10    1:4    26
496  10    1:4    26
497  10    1:4    26    (* READ IN THE FILE *)
498  10    1:4    26
499  10    1:3    26    READFILE;
500  10    1:3    28    1: IF (EBUF^[BUFCOUNT-1]<>CHR(EOL)) OR (BUFCOUNT=1) THEN
501  10    1:4    42      BEGIN
502  10    1:5    42        EBUF^[BUFCOUNT]:=CHR(EOL);
503  10    1:5    47        BUFCOUNT:=BUFCOUNT+1;
504  10    1:4    52      END;
505  10    1:4    52
506  10    1:4    52
507  10    1:4    52    (* INITIALIZE EVERYTHING ELSE! *)
508  10    1:4    52
509  10    1:3    52    DIRECTION:='>';
510  10    1:3    55    LASTPAT:=1; (* INIT TO THE BEGINNING OF THE BUFFER (FOR EQUALC) *)
511  10    1:3    58    COPYOK:=FALSE;
512  10    1:3    62    LINE1PTR:=1;
513  10    1:3    66    (* THESE DO NOT YET GO THROUGH THE SCREEN AND KEYBOARD CONTROL
514  10    1:3    66       RECORDS *)
515  10    1:3    66    WITH SYSCOM^.CRTINFO DO
516  10    1:4    74      BEGIN
517  10    1:5    74        ESC:=ORD(ALTMODE);
518  10    1:5    85        BS:=ORD(CHARDEL);
519  10    1:5    96        DEL:=ORD(LINEDEL);
520  10    1:5    07        SCREENWIDTH:=WIDTH-1;
521  10    1:5    15        SCREENHEIGHT:=HEIGHT-1;
522  10    1:5    23        MIDDLE:=(SCREENHEIGHT DIV 2) + 1;
523  10    1:4    31      END;
524  10    1:3    31    ETX:=ORD(KEYBRD.CHRETXKEY]); [CHANGED FROM SYSCOM ASSIGNMENT 11/2/78 MAB]
525  10    1:3    39    MAP(CHR(BS),LEFT);                    (* MAP BACKSPACE KEY FOR NOW *)
526  10    1:3    45    SYSCOM^.MISCINFO.NOBREAK := TRUE;
527  10    1:3    54    [INCLUDING THE COMMAND PROMPT LINE]
528  10    1:3    54    DEFPROMPTS;
529  10    1:3    56    SDEFINED:=FALSE; TDEFINED:=FALSE; (* NO SUBSTRING OR TARGET *)
530  10    1:3    64    WITH PAGEZERO DO
531  10    1:4    64      IF NOT DEFINED THEN
```

```
532  10    1:5     70            BEGIN
533  10    1:6     70              FILLCHAR(BUF,1024,CHR(0));
534  10    1:6     80              CREATED:=THEDATE; LASTUSED:=THEDATE;
535  10    1:6     96              TOKDEF:=TRUE; (* DEFAULT MODE IS T(OKEN *)
536  10    1:6     00              FILLING:=FALSE; AUTOINDENT:=TRUE; RUNOFFCH:='^';
537  10    1:6     12              LMARGIN:=0; PARAMARGIN:=5; RMARGIN:=SCREENWIDTH;
538  10    1:6     24              DEFINED:=TRUE;
539  10    1:5     28            END;
540  10    1:2     28      END(* WITH *);
541  10    1:2     28
542  10    1:2     28
543  10    1:2     28    (* INITIALIZE THE KIND ARRAY FOR TOKEN FIND *)
544  10    1:2     28
545  10    1:1     28    FOR CH:=CHR(0) TO CHR(255) DO KIND[CH]:=ORD(CH);  (* MAKE THEM ALL UNIQUE *)
546  10    1:1     58    FOR CH:='A' TO 'Z' DO KIND[CH]:=ORD('A');
547  10    1:1     86    FOR CH:='A' TO 'Z' DO KIND[CH]:=ORD('A');
548  10    1:1   1 14    FOR CH:='0' TO '9' DO KIND[CH]:=ORD('A');
549  10    1:1   1 42    KIND[CHR(EOL)]:=ORD(' '); KIND[CHR(HT)] :=ORD(' ');
550  10    1:1   1 56    FILLCHAR(BLANKAREA,SIZEOF(BLANKAREA),' ');
551  10    1:1   1 66    SAVETOP:='';
552  10    1:1   1 74
553  10    1:0   1 74  END(* INITIALIZE *);
554  10    1:0   1 06
555  10    1:0   1 06
556  10    1:0   1 06  (*$I INIT       *)
556  10    1:0   1 06  (*$I OUT        *)
557  11    1:D     3  SEGMENT FUNCTION OUT: BOOLEAN;
558  11    1:D     3  LABEL 1,2;
559  11    1:D     3  VAR
560  11    1:D     3    SAVE: PTRTYPE;
561  11    1:D     4    I: INTEGER;
562  11    1:D     5    BUF: PACKED ARRAY [0..1023] OF CHAR;
563  11    1:D    17    FN: STRING;
564  11    1:0     0  BEGIN
565  11    1:1     0    OUT:=FALSE;
566  11    1:1     3    REPEAT
567  11    1:2     3      CLEARSCREEN;      (* DUMB TERMINAL PATCH *)
568  11    1:2     6      SAVETOP:='>QUIT:';
569  11    1:2    20      WRITELN(SAVETOP);
570  11    1:2    36      WRITELN('     U(PDATE THE WORKFILE AND LEAVE');
571  11    1:2    87      WRITELN('     E(XIT WITHOUT UPDATING');
```

```
572  11    1:2    30      WRITELN('      R(ETURN TO THE EDITOR WITHOUT UPDATING');
573  11    1:2    89      WRITELN('      W(RITE TO A FILE NAME AND RETURN');
574  11    1:2    42      CH:=UCLC(GETCH);
575  11    1:1    54    UNTIL CH IN ['U','E','R','W'];
576  11    1:1    74    IF CH='R' THEN GOTO 2;
577  11    1:1    81    IF CH='E' THEN BEGIN OUT:=TRUE; CLEARSCREEN; GOTO 2 END;
578  11    1:1    94    CLOSE(THEFILE);
579  11    1:1    01    IF CH='W' THEN
580  11    1:2    06      BEGIN
581  11    1:3    06        SAVE:=CURSOR;
582  11    1:3    09        BLANKCRT(1);
583  11    1:3    13        WRITE('NAME OF OUTPUT FILE (<CR> TO RETURN)  -->');
584  11    1:3    64        READLN(FN);
585  11    1:3    80        IF LENGTH(FN)=0 THEN GOTO 2;
586  11    1:3    91        FOR I:=1 TO LENGTH(FN) DO FN[I]:=UCLC(FN[I]);
587  11    1:3    31        IF ((POS('.TEXT',FN)<>LENGTH(FN)-4) OR (LENGTH(FN)<=4)) AND
588  11    1:3    63           (FN[LENGTH(FN)]<>'.') THEN
589  11    1:4    77          FN:=CONCAT(FN,'.TEXT');
590  11    1:3    14        IF FN[LENGTH(FN)]='.' THEN DELETE(FN,LENGTH(FN),1);
591  11    1:2    39      END
592  11    1:1    39    ELSE
593  11    1:2    41      FN:='*SYSTEM.WRK.TEXT';
594  11    1:1    65    BLANKCRT(1);
595  11    1:1    69    WRITE('WRITING');
596  11    1:1    86    OPENNEW(THEFILE,FN);
597  11    1:1    97    PAGEZERO.LASTUSED:=THEDATE;
598  11    1:1    05    IF BLOCKWRITE(THEFILE,PAGEZERO,2) <> 2 THEN GOTO 1;
599  11    1:1    29    WRITE('.');
600  11    1:1    37    CURSOR:=1;
601  11    1:1    40    WHILE CURSOR < BUFCOUNT-1023 DO
602  11    1:2    49      BEGIN
603  11    1:3    49        I:=SCAN(-1022,=CHR(EOL),EBUF^[CURSOR+1022]);
604  11    1:3    67        MOVELEFT(EBUF^[CURSOR],BUF,1023+I);
605  11    1:3    80        FILLCHAR(BUF[1023+I],ABS(I)+1,CHR(0));
606  11    1:3    94        IF BLOCKWRITE(THEFILE,BUF,2) <> 2 THEN GOTO 1;
607  11    1:3    17        CURSOR:=CURSOR+1023+I;
608  11    1:3    26        WRITE('.');
609  11    1:2    34      END;
610  11    1:1    36    IF CURSOR<BUFCOUNT THEN
611  11    1:2    41      BEGIN
612  11    1:3    41        FILLCHAR(BUF,SIZEOF(BUF),CHR(0));
```

```
613  11    1:3    50       MOVELEFT(EBUF^[CURSOR],BUF,BUFCOUNT-CURSOR);
614  11    1:3    61       IF BLOCKWRITE(THEFILE,BUF,2) <> 2 THEN GOTO 1; WRITE('.')
615  11    1:2    92     END;
616  11    1:1    92   CLOSE(THEFILE,LOCK);
617  11    1:1    99   WRITELN;
618  11    1:1    05   WRITELN('YOUR FILE IS ',BUFCOUNT,' BYTES LONG.');
619  11    1:1    64   IF CH='U' THEN
620  11    1:2    69     WITH USERINFO DO
621  11    1:3    69       BEGIN
622  11    1:4    69         SYMVID:=SYVID; SYMTID:='SYSTEM.WRK.TEXT'; GOTSYM:=TRUE;
623  11    1:4    04         OPENOLD(THEFILE,'*SYSTEM.WRK.CODE'); CLOSE(THEFILE,PURGE);
624  11    1:4    38         GOTCODE:=FALSE; CODETID:=''; OUT:=TRUE;
625  11    1:3    53       END
626  11    1:1    53     ELSE
627  11    1:2    55       BEGIN
628  11    1:3    55         WRITE('DO YOU WANT TO E(XIT FROM OR R(ETURN TO THE EDITOR? ');
629  11    1:3  1 17         REPEAT CH:=UCLC(GETCH) UNTIL CH IN ['E','R'];
630  11    1:3  1 48         OUT:= CH='E';
631  11    1:3  1 53         CURSOR:=SAVE; (* QW RETURNS TO THE EDITOR *)
632  11    1:2  1 56       END;
633  11    1:1  1 56   GOTO 2; (* SORRY ABOUT THAT EDSGER *)
634  11    1:1  1 58 1: ERROR('WRITING OUT THE FILE',NONFATAL);
635  11    1:1  1 85 2:END;
636  11    1:1  1 12
637  11    1:1  1 12
638  11    1:1  1 12
639  11    1:1  1 12 (*$I OUT        *)
639  11    1:1  1 12 (*$I COPYFILE   *)
640  12    1:D    1 SEGMENT PROCEDURE COPYFILE;
641  12    1:D    1 VAR
642  12    1:D    1   STARTPAGE,STOPPAGE,STARTOFFSET,STOPOFFSET,
643  12    1:D    1   LEFTPART,PAGE,NOTNULLS,THEREST,LMOVE: INTEGER;
644  12    1:D   10   DONE,OVFLW: BOOLEAN;
645  12    1:D   12   BUFR: PACKED ARRAY [0..1023] OF CHAR;
646  12    1:D   24   STARTMARK,STOPMARK: PACKED ARRAY [0..7] OF CHAR;
647  12    1:D   32   FN: STRING;
648  12    1:D   73   F: FILE;
649  12    1:D   13
650  12    2:D    1 PROCEDURE ERRMARKER;
651  12    2:0    0 BEGIN
652  12    2:1    0   ERROR('IMPROPER MARKER SPECIFICATION.',NONFATAL);
```

```
653  12    2:1    37   EXIT(COPYFILE)
654  12    2:0    41 END;
655  12    2:0    54
656  12    3:D     1 PROCEDURE UNSPLITBUF;
657  12    3:D     1 (* STICH THE BUFFER BACK TOGETHER AGAIN. *)
658  12    3:0     0 BEGIN
659  12    3:1     0   MOVELEFT(EBUF^[THEREST],EBUF^[CURSOR],LMOVE);
660  12    3:1    13   READJUST(LEFTPART+1,CURSOR-(LEFTPART+1));
661  12    3:1    28   BUFCOUNT:=BUFCOUNT+CURSOR-(LEFTPART+1);
662  12    3:1    39   CURSOR:=LEFTPART+1; (* CURSOR POINTS TO THE BEGINNING OF THE FILE *)
663  12    3:0    46 END;
664  12    3:0    58
665  12    4:D     1 PROCEDURE READERR;
666  12    4:0     0 BEGIN
667  12    4:1     0   ERROR('MARKER EXCEEDS FILE BOUNDS.',NONFATAL);
668  12    4:1    34   UNSPLITBUFF;
669  12    4:1    36   CENTERCURSOR(TRASH,MIDDLE,TRUE);
670  12    4:1    46   EXIT(COPYFILE)
671  12    4:0    50 END;
672  12    4:0    62
673  12    5:D     1 PROCEDURE SPLITBUF;
674  12    5:D     1 (* SPLIT THE BUFFER AT THE CURSOR.  THEREST POINTS TO THE RIGHT PART, LMOVE
675  12    5:D     1    IS THE LENGTH OF THE RIGHT PART, LEFTPART POINTS TO THE END OF THE 'LEFT
676  12    5:D     1    PART', AND CURSOR REMAINS UNCHANGED. *)
677  12    5:0     0 BEGIN
678  12    5:1     0   THEREST:=BUFSIZE-(BUFCOUNT-CURSOR);
679  12    5:1     8   LMOVE:=BUFCOUNT-CURSOR+1;
680  12    5:1    16   LEFTPART:=CURSOR-1;
681  12    5:1    22   MOVERIGHT(EBUF^[CURSOR],EBUF^[THEREST],LMOVE)
682  12    5:0    35 END;
683  12    5:0    48
684  12    6:D     1 PROCEDURE PARSEFN;
685  12    6:D     1 VAR I,LPTR,RPTR,COMMA: INTEGER;
686  12    6:D     5     MARK: STRING;
687  12    6:0     0 BEGIN
688  12    6:1     0   LPTR:=POS('[',FN);
689  12    6:1    15   IF LPTR=0 THEN
690  12    6:2    20     BEGIN (* WHOLE FILE *)
691  12    6:3    20       STARTMARK:='        ';
692  12    6:3    37       STOPMARK:= '        '
693  12    6:2    41     END
```

```
694  12   6:1   54    ELSE
695  12   6:2   56      BEGIN
696  12   6:3   56        RPTR:=POS(']',FN);
697  12   6:3   71        IF (RPTR=0) OR (RPTR<LPTR) OR (RPTR<>LENGTH(FN)) THEN ERRMARKER;
698  12   6:3   91        MARK:=COPY(FN,LPTR+1,RPTR-LPTR-1); (* STUFF BETWEEN THE BRACKETS *)
699  12   6:3   14        FN:=COPY(FN,1,LPTR-1);
700  12   6:3   35        COMMA:=POS(',',MARK);
701  12   6:3   48        IF COMMA=0 THEN ERRMARKER;
702  12   6:3   55        I:=LENGTH(MARK)-COMMA; (* SECOND MARKER PTR *)
703  12   6:3   63        MOVELEFT(MARK[1],STARTMARK,MIN(8,COMMA-1));
704  12   6:3   82        FILLCHAR(STARTMARK[COMMA-1],MAX(0,8-(COMMA-1)),' ');
705  12   6:3   03        MOVELEFT(MARK[COMMA+1],STOPMARK,MIN(I,8));
706  12   6:3   22        FILLCHAR(STOPMARK[I],MAX(0,8-I),' ')
707  12   6:2   39      END;
708  12   6:1   39    FOR I:=0 TO 7 DO STARTMARK[I]:=UCLC(STARTMARK[I]);
709  12   6:1   75    FOR I:=0 TO 7 DO STOPMARK [I]:=UCLC(STOPMARK[I]);
710  12   6:1   11    FOR I:=1 TO LENGTH(FN) DO FN[I]:=UCLC(FN[I]);
711  12   6:1   52    IF ((POS('.TEXT',FN)<>LENGTH(FN)-4) OR
712  12   6:1   78       (LENGTH(FN)<=4)) AND (FN[LENGTH(FN)]<>'.') THEN
713  12   6:2   03      FN:=CONCAT(FN,'.TEXT');
714  12   6:1   38    IF FN[LENGTH(FN)]='.' THEN DELETE(FN,LENGTH(FN),1);
715  12   6:0   67  END;
716  12   6:0   88
717  12   7:D    1  PROCEDURE STUFFIT(START,STOP:INTEGER);
718  12   7:D    3  (* PUT THE CONTENTS OF BUFR INTO EBUF.  OVFLW IS SET TO TRUE WHEN THERE IS
719  12   7:D    3     NO MORE ROOM IN THE BUFFER. *)
720  12   7:D    3  VAR AMOUNT: INTEGER;
721  12   7:0    0  BEGIN
722  12   7:1    0    IF START<=STOP THEN
723  12   7:2    5      BEGIN
724  12   7:3    5        AMOUNT:=STOP-START+1;
725  12   7:3   12        IF CURSOR+AMOUNT+250(*SLOP*)>=THEREST THEN
726  12   7:4   25          BEGIN
727  12   7:5   25            ERROR('BUFFER OVERFLOW.',NONFATAL);
728  12   7:5   48            UNSPLITBUFF;
729  12   7:5   50            CENTERCURSOR(TRASH,MIDDLE,TRUE);
730  12   7:5   60            EXIT(COPYFILE)
731  12   7:4   64          END
732  12   7:3   64        ELSE
733  12   7:4   66          BEGIN
734  12   7:5   66            MOVELEFT(BUFR[START],EBUF^[CURSOR],AMOUNT);
```

```
735  12    7:5    76            CURSOR:=CURSOR+AMOUNT
736  12    7:4    77          END
737  12    7:2    81      END
738  12    7:0    81  END;
739  12    7:0    94
740  12    8:D     1  PROCEDURE GETNEXT;
741  12    8:0     0  BEGIN
742  12    8:1     0    DONE:=BLOCKREAD(F,BUFR,2,PAGE+PAGE)<>2;
743  12    8:1    27    WRITE('.');
744  12    8:1    35    IF NOT DONE THEN NOTNULLS:=SCAN(-1024,<>CHR(0),BUFR[1023])+1024
745  12    8:1    56    ELSE NOTNULLS:=0;
746  12    8:1    69    PAGE:=PAGE+1;
747  12    8:0    77  END;
748  12    8:0    90
749  12    9:D     1  PROCEDURE CHKOVFLW;
750  12    9:0     0  BEGIN
751  12    9:1     0    IF (STOPOFFSET>=NOTNULLS) AND (STOPPAGE<PAGE) THEN
752  12    9:2    17      BEGIN
753  12    9:3    17        STOPPAGE:=STOPPAGE+1;
754  12    9:3    25        STOPOFFSET:=STOPOFFSET-NOTNULLS;
755  12    9:2    35      END;
756  12    9:0    35  END;
757  12    9:0    48
758  12   10:D     1  PROCEDURE FINDMARKERS;
759  12   10:D     1  (* GIVEN STARTMARK AND STOPMARK FIND OUT THEIR PAGE NUMBERS AND OFFSETS *)
760  12   10:D     1  VAR
761  12   10:D     1    PZ: HEADER;
762  12   10:D    13
763  12   11:D     1    PROCEDURE SEARCH(MNAME:NAME;VAR OFF,PNUM: INTEGER);
764  12   11:D     8    VAR
765  12   11:D     8      I: INTEGER;
766  12   11:0     0    BEGIN
767  12   11:1     0      I:=0;
768  12   11:1     8      WHILE (I<PZ.COUNT) AND (MNAME<>PZ.NAME[I]) DO I:=I+1;
769  12   11:1    34      IF MNAME<>PZ.NAME[I] THEN
770  12   11:2    47        BEGIN
771  12   11:3    47          ERROR('MARKER NOT THERE.',NONFATAL);
772  12   11:3    71          UNSPLITBUFF;
773  12   11:3    73          EXIT(COPYFILE)
774  12   11:2    77        END;
775  12   11:1    77      OFF:=PZ.POFFSET[I];
```

```
776  12   11:1    86     PNUM:=PZ.PAGENDEX[I];
777  12   11:1    93     IF PNUM=0 THEN
778  12   11:2    99       BEGIN OFF:=OFF-1; PNUM:=1 END; (* KLUDGE TO MAINTAIN COMPATIBILITY *)
779  12   11:0    08   END;
780  12   11:0    22
781  12   10:0     0 BEGIN(* FINDMARKERS *)
782  12   10:1     0   STARTPAGE:=1;    STARTOFFSET:=0;   (* DEFAULT VALUES *)
783  12   10:1     8   STOPPAGE:=32767; STOPOFFSET:=32767;
784  12   10:1    20   IF (STARTMARK<>'         ') OR (STOPMARK<>'         ') THEN
785  12   10:2    59     BEGIN
786  12   10:3    59       IF BLOCKREAD(F,PZ,2,0)<>2 THEN READERR;
787  12   10:3    80       IF STARTMARK<>'         ' THEN SEARCH(STARTMARK,STARTOFFSET,STARTPAGE);
788  12   10:3    12       IF STOPMARK<>'         ' THEN SEARCH(STOPMARK,STOPOFFSET,STOPPAGE)
789  12   10:2    42     END
790  12   10:0    44 END;
791  12   10:0    56
792  12    1:0     0 BEGIN
793  12    1:1     0   PROMPTLINE:=' COPY: FROM WHAT FILE[MARKER,MARKER]? ';
794  12    1:1    59   REPEAT
795  12    1:2    59     PROMPT;
796  12    1:2    62     READLN(FN);
797  12    1:2    78     IF LENGTH(FN)=0 THEN EXIT(COPYFILE);
798  12    1:2    91     PARSEFN;
799  12    1:2    93     RESET(F,FN);
800  12    1:2    04     PROMPTLINE:=' COPY: FILE NOT PRESENT, FILENAME? ';
801  12    1:1    47   UNTIL IORESULT=0;
802  12    1:1    53   PROMPTLINE:=' COPY'; PROMPT;
803  12    1:1    69   SPLITBUF;
804  12    1:1    71   FINDMARKERS;
805  12    1:1    73   PAGE:=STARTPAGE;
806  12    1:1    76   GETNEXT;
807  12    1:1    78   WHILE (STARTOFFSET>=NOTNULLS) AND NOT DONE DO
808  12    1:2    86     BEGIN
809  12    1:3    86       CHKOVFLW;
810  12    1:3    88       STARTOFFSET:=STARTOFFSET-NOTNULLS;
811  12    1:3    93       GETNEXT;
812  12    1:2    95     END;
813  12    1:1    97   IF (STOPPAGE<PAGE) AND (STOPOFFSET<NOTNULLS) THEN
814  12    1:2    06     STUFFIT(STARTOFFSET,MIN(NOTNULLS-1,STOPOFFSET-1))
815  12    1:1    18   ELSE
816  12    1:2    22     STUFFIT(STARTOFFSET,NOTNULLS-1);
```

```
817  12    1:1    28    WHILE ((STOPPAGE>=PAGE) OR (STOPOFFSET>=NOTNULLS)) AND NOT DONE DO
818  12    1:2    40      BEGIN
819  12    1:3    40        CH<OVFLW;
820  12    1:3    42        GETNEXT;
821  12    1:3    44        IF (STOPPAGE<PAGE) AND (STOPOFFSET<NOTNULLS) THEN
822  12    1:4    53          STUFFIT(0,MIN(NOTNULLS-1,STOPOFFSET-1))
823  12    1:3    65        ELSE
824  12    1:4    69          STUFFIT(0,NOTNULLS-1)
825  12    1:2    73      END;
826  12    1:1    77    IF IORESULT<>0 THEN ERROR('DISK ERROR.',NONFATAL);
827  12    1:1    01    UNSPLITBUF;
828  12    1:1    03    CENTERCURSOR(TRASH,MIDDLE,TRUE);
829  12    1:1    13    CLOSE(F);
830  12    1:0    20  END;
831  12    1:0    46
832  12    1:0    46  (*$I COPYFILE  *)
832  12    1:0    46  (*$I ENVIRON   *)
833  13    1:D     1  SEGMENT PROCEDURE ENVIRONMENT;
834  13    1:D     1  VAR
835  13    1:D     1    I: INTEGER;
836  13    1:D     2
837  13    2:D     1    PROCEDURE ERASE10;
838  13    2:D     1    VAR I: INTEGER;
839  13    2:0     0    BEGIN
840  13    2:1     0      WRITE(' ':10);
841  13    2:1     8      FOR I:=1 TO 10 DO WRITE(CHR(BS));
842  13    2:0    36    END;
843  13    2:0    50
844  13    3:D     1    PROCEDURE BOOL(B:BOOLEAN);
845  13    3:0     0    BEGIN
846  13    3:1     0      IF B THEN WRITE('TRUE') ELSE WRITE('FALSE');
847  13    3:1    34      WRITELN
848  13    3:0    34    END;
849  13    3:0    52
850  13    4:D     3    FUNCTION GETBOOL: BOOLEAN;
851  13    4:D     3    VAR CH: CHAR;
852  13    4:0     0    BEGIN
853  13    4:1     0      ERASE10; CH:=UCLC(GETCH);
854  13    4:1    14      WHILE NOT (CH IN ['T','F']) DO
855  13    4:2    35        BEGIN
856  13    4:3    35          WRITE('T OR F');
```

```
857  13    4:3    51          FOR TRASH:=0 TO 5 DO WRITE(CHR(BS));
858  13    4:3    85          CH:=UCLC(GETCH)
859  13    4:2    90        END;
860  13    4:1    99      IF CH='T' THEN
861  13    4:2    04        BEGIN
862  13    4:3    04          WRITE('TRUE  ');
863  13    4:3    20          GETBOOL:=TRUE
864  13    4:2    20        END
865  13    4:1    23      ELSE
866  13    4:2    25        BEGIN
867  13    4:3    25          WRITE('FALSE ');
868  13    4:3    41          GETBOOL:=FALSE
869  13    4:2    41        END;
870  13    4:0    44    END;
871  13    4:0    60
872  13    5:D     3    FUNCTION GETINT: INTEGER;
873  13    5:D     3    VAR
874  13    5:D     3      CH:CHAR;
875  13    5:D     4      N: INTEGER;
876  13    5:0     0    BEGIN
877  13    5:1     0      ERASE10;
878  13    5:1     2      N:=0;
879  13    5:1     5      REPEAT
880  13    5:2     5        REPEAT
881  13    5:3     5          CH:=GETCH;
882  13    5:3    12          IF NOT (CH IN ['0'..'9',CHR(SP),CHR(CR)])
883  13    5:3    31            THEN WRITE('#',CHR(BELL),CHR(BS));
884  13    5:2    60        UNTIL CH IN ['0'..'9',CHR(SP),CHR(CR)];
885  13    5:2    81        IF CH IN ['0'..'9'] THEN
886  13    5:3    96          BEGIN
887  13    5:4    96            WRITE(CH);
888  13    5:4    04            IF N<1000 THEN N:=N*10+ORD(CH)-ORD('0')
889  13    5:3    17          END;
890  13    5:1    20      UNTIL CH IN [CHR(SP),CHR(CR)];
891  13    5:1    29      GETINT:=N; WRITE('  ')
892  13    5:0    44    END;
893  13    5:0    60
894  13    1:0     0  BEGIN
895  13    1:1     0    WITH PAGEZERO DO
896  13    1:2     0      BEGIN
897  13    1:3     0        CLEARSCREEN;
```

```
898  13    1:3     3      PROMPTLINE:= ' ENVIRONMENT: [OPTIONS] <ETX> OR <SP> TO LEAVE';
899  13    1:3    57      PROMPT; NEEDPROMPT:=TRUE;
900  13    1:3    64      WRITELN;
901  13    1:3    70      WRITE(        '    A(UTO INDENT  '); BOOL(AUTOINDENT);
902  13    1:3    03      WRITE(        '    F(ILLING      '); BOOL(FILLING);
903  13    1:3    36      WRITE(        '    L(EFT MARGIN  '); WRITELN(LMARGIN);
904  13    1:3    80      WRITE(        '    R(IGHT MARGIN '); WRITELN(RMARGIN);
905  13    1:3    24      WRITE(        '    P(ARA MARGIN  '); WRITELN(PARAMARGIN);
906  13    1:3    68      WRITE(        '    C(OMMAND CH   '); WRITELN(RUNOFFCH);
907  13    1:3    12      WRITE(        '    T(OKEN DEF    '); BOOL(TOKDEF);
908  13    1:3    45      WRITELN;
909  13    1:3    51      WRITELN('    ',BUFCOUNT,' BYTES USED, ',BUFSIZE-BUFCOUNT+1,' AVAILABLE.');
910  13    1:3    35      WRITELN;
911  13    1:3    41      IF SDEFINED OR TDEFINED THEN
912  13    1:4    50        BEGIN
913  13    1:5    50          WRITELN('    PATTERNS:');
914  13    1:5    79          IF TDEFINED THEN WRITE('      <TARGET>= ''',TARGET:TLENGTH,'''');
915  13    1:5    34          IF SDEFINED THEN WRITE(', <SUBST>=  ''',SUBSTRING:SLENGTH,'''');
916  13    1:5    85          WRITELN; WRITELN;
917  13    1:4    97        END;
918  13    1:3    97      IF COUNT>0 THEN WRITELN('    MARKERS:');
919  13    1:3    32      WRITE('  ');
920  13    1:3    44      FOR I:=0 TO COUNT-1 DO
921  13    1:4    59        BEGIN WRITE(' ':6,NAME[I]);
922  13    1:5    81          IF (I+4) MOD 3=0 THEN BEGIN WRITELN; WRITE('  ') END
923  13    1:4    08        END;
924  13    1:3    15      WRITELN;
925  13    1:3    21      WRITELN;
926  13    1:3    27      WRITELN('    DATE CREATED: ',CREATED.MONTH,'-',CREATED.DAY,'-',
927  13    1:3    97                                 CREATED.YEAR,'   LAST USED: ',
928  13    1:3    34                                 LASTUSED.MONTH,'-',LASTUSED.DAY,'-',
929  13    1:3    76                                 LASTUSED.YEAR);
930  13    1:3    95      GOTOXY(LENGTH(PROMPTLINE),0);
931  13    1:3    04      REPEAT
932  13    1:4    04        CH:=UCLC(GETCH);
933  13    1:4    16        IF NOT (CH IN ['A','C','F','L','P','R','T',' ',CHR(ETX),CHR(CR)]) THEN
934  13    1:5    44          BEGIN ERROR('NOT OPTION',NONFATAL); PROMPT; END
935  13    1:4    64        ELSE
936  13    1:5    66          CASE CH OF
937  13    1:5    69            'A': BEGIN GOTOXY(18,1); AUTOINDENT:=GETBOOL END;
938  13    1:5    83            'F': BEGIN GOTOXY(18,2); FILLING:=GETBOOL END;
```

```
939  13    1:5     97            'L': BEGIN GOTOXY(18,3); LMARGIN:=GETINT END;
940  13    1:5   1 11            'R': BEGIN GOTOXY(18,4); RMARGIN:=GETINT END;
941  13    1:5   1 25            'P': BEGIN GOTOXY(18,5); PARAMARGIN:=GETINT END;
942  13    1:5   1 39            'C': BEGIN GOTOXY(18,6); READ(RUNOFFCH) END;
943  13    1:5   1 55            'T': BEGIN GOTOXY(18,7); TOKDEF:=GETBOOL END
944  13    1:5   1 67          END;
945  13    1:4   1 16        GOTOXY(LENGTH(PROMPTLINE),0);
946  13    1:3   1 25      UNTIL CH IN [' ',CHR(ETX),CHR(CR)];
947  13    1:3   1 47      REDISPLAY;
948  13    1:2   1 50    END;
949  13    1:0   1 50 END;
950  13    1:0   1 72
951  13    1:0   1 72
952  13    1:0   1 72
953  13    1:0   1 72 (*$I ENVIRON   *)
953  13    1:0   1 72 (*$I PUTSYNTAX *)
954  14    1:D     1 SEGMENT PROCEDURE PUTSYNTAX;
955  14    1:D     1 VAR
956  14    1:D     1   D0,D1,D2,BLK,PTR,COLON: INTEGER;
957  14    1:D     7   T,C:PACKED ARRAY [0..2] OF CHAR;
958  14    1:D    11   BUF:PACKED ARRAY [0..1023] OF CHAR;
959  14    1:D    23   F: FILE;
960  14    1:D    63
961  14    2:D     1 PROCEDURE PUTNUM;
962  14    2:0     0 BEGIN
963  14    2:1     0   MSG:='SYNTAX ERROR #'; PUTMSG;
964  14    2:1    25   WRITE(USERINFO.ERRNUM,'. TYPE <SP>');
965  14    2:0    56 END;
966  14    2:0    68
967  14    1:0     0 BEGIN (* PUTSYNTAX *)
968  14    1:1     0   WITH USERINFO DO
969  14    1:2    13     BEGIN
970  14    1:3    13       OPENOLD(F,'*SYSTEM.SYNTAX');
971  14    1:3    38       IF IORESULT<>0 THEN PUTNUM
972  14    1:3    44       ELSE
973  14    1:4    48         BEGIN
974  14    1:5    48           IF ERRNUM<=104 THEN BLK:=2
975  14    1:5    55           ELSE
976  14    1:6    60             IF ERRNUM<=126 THEN BLK:=4
977  14    1:6    67             ELSE
978  14    1:7    72               IF ERRNUM<=151 THEN BLK:=6
```

```
 979  14    1:7   81                ELSE
 980  14    1:8   86                  IF ERRNUM<=185 THEN BLK:=8
 981  14    1:8   95                  ELSE
 982  14    1:9   00                    IF ERRNUM<=302 THEN BLK:=10
 983  14    1:9   09                    ELSE BLK:=12;
 984  14    1:5   17              IF BLOCKREAD(F,BUF,2,BLK)<>2 THEN PUTNUM
 985  14    1:5   35              ELSE
 986  14    1:6   39                BEGIN
 987  14    1:7   39                  IF BUF[0]=CHR(DLE) THEN PTR:=2 ELSE PTR:=0;
 988  14    1:7   55                  D0:=ERRNUM DIV 100; (* CONVERT ERROR NUMBER TO CHARACTERS *)
 989  14    1:7   62                  D1:=(ERRNUM-D0*100) DIV 10;
 990  14    1:7   73                  D2:=ERRNUM MOD 10;
 991  14    1:7   80                  T[0]:=CHR(D0+ORD('0')); T[1]:=CHR(D1+ORD('0'));
 992  14    1:7   94                  T[2]:=CHR(D2+ORD('0'));
 993  14    1:7   01                  REPEAT
 994  14    1:8   01                    FILLCHAR(C,3,'0');
 995  14    1:8   08                    COLON:=SCAN(MAXCHAR,=':',BUF[PTR]);
 996  14    1:8   21                    MOVELEFT(BUF[PTR],C[3-COLON],COLON);
 997  14    1:8   32                    COLON:=COLON+PTR;
 998  14    1:8   37                    PTR:=SCAN(MAXCHAR,=CHR(EOL),BUF[PTR])+PTR+3
 999  14    1:7   50                  UNTIL (T=C) OR (BUF[PTR]=CHR(0));
1000  14    1:7   70                  IF (T<>C) AND (BUF[PTR]=CHR(0)) THEN PUTNUM
1001  14    1:7   86                  ELSE
1002  14    1:8   90                    BEGIN
1003  14    1:9   90                      MOVELEFT(BUF[COLON+1],MSG[1],(PTR-COLON)-4);
1004  14    1:9   06                      MSG[0]:=CHR(MIN(68,(PTR-COLON)-4)); (* R- REQUIRED *)
1005  14    1:9   22                      HOME; CLEARLINE(0); WRITE(MSG,'.  TYPE <SP>');
1006  14    1:8   61                    END
1007  14    1:6   61                END
1008  14    1:4   61              END(* IF IORESULT<>0 *);
1009  14    1:3   61            SHOWCURSOR;
1010  14    1:3   64            REPEAT UNTIL GETCH=' ';
1011  14    1:3   73            ERRBLK:=0; ERRSYM:=0; ERRNUM:=0; (*  ONLY YELL ONCE!!! *)
1012  14    1:2   85          END(* WITH USERINFO *)
1013  14    1:0   85 END(* PUTSYNTAX *);
1014  14    1:0   12
1015  14    1:0   12
1016  14    1:0   12 (*$I PUTSYNTAX *)
1016  14    1:0   12 (*$I COMMAND   *)
1017  15    1:D    1 SEGMENT PROCEDURE EDITCORE;
1018  15    1:D    1
```

```
1019  15    1:0     1 (* CORE PROCEDURES.  EXECUTE THESE COMMANDS UNTIL EITHER A SET ENVIRONMENT
1020  15    1:0     1    COMES ALONG OR A QUIT COMMAND. *)
1021  15    1:0     1
1022  15    1:0     1
1023  15    1:0     1
1024  15    2:0     1 PROCEDURE NEXTCOMMAND; FORWARD;
1025  15    2:0     1
1026  15    3:0     1 PROCEDURE FIXDIRECTION;
1027  15    3:0     0 BEGIN
1028  15    3:1     0   IF COMMAND=FORWARDC THEN DIRECTION:='>' ELSE DIRECTION:='<';
1029  15    3:1    13   HOME; WRITE(DIRECTION); (* UPDATE PROMPT LINE *)
1030  15    3:1    24   SHOWCURSOR; NEXTCOMMAND
1031  15    3:0    27 END;
1032  15    3:0    42
1033  15    4:0     1 PROCEDURE COPY;
1034  15    4:0     0 BEGIN
1035  15    4:1     0   PROMPTLINE:=' COPY: B(UFFER  F(ROM FILE  <ESC>';
1036  15    4:1    41   PROMPT; NEEDPROMPT:=TRUE;
1037  15    4:1    48   REPEAT
1038  15    4:2    48     CH:=UCLC(GETCH);
1039  15    4:1    60   UNTIL CH IN ['B','F',CHR(ESC)];
1040  15    4:1    83   IF CH='B' THEN
1041  15    4:2    88     BEGIN
1042  15    4:3    88       IF NOT COPYOK OR ((BUFCOUNT+COPYLENGTH+10>COPYSTART)
1043  15    4:3    03                         AND (COPYSTART>=BUFCOUNT))
1044  15    4:3    09         THEN ERROR('INVALID COPY.',NONFATAL)
1045  15    4:3    29         ELSE
1046  15    4:4    34           IF BUFCOUNT+COPYLENGTH>=BUFSIZE THEN ERROR('NO ROOM',NONFATAL)
1047  15    4:4    54           ELSE
1048  15    4:5    59             BEGIN
1049  15    4:6    59               IF COPYLINE THEN
1050  15    4:7    64                 BEGIN
1051  15    4:8    64                   GETLEADING;
1052  15    4:8    67                   CURSOR:=LINESTART
1053  15    4:7    67                 END;
1054  15    4:6    70               MOVERIGHT(EBUF^[CURSOR],EBUF^[CURSOR+COPYLENGTH],BUFCOUNT-CURSOR+1);
1055  15    4:6    87               IF (COPYSTART>=CURSOR) AND (COPYSTART<BUFCOUNT) THEN
1056  15    4:7    00                 MOVELEFT(EBUF^[COPYSTART+COPYLENGTH],EBUF^[CURSOR],COPYLENGTH)
1057  15    4:6    17               ELSE
1058  15    4:7    19                 MOVELEFT(EBUF^[COPYSTART],EBUF^[CURSOR],COPYLENGTH);
1059  15    4:6    32               BUFCOUNT:=BUFCOUNT+COPYLENGTH;
```

```
1060  15    4:6    39                READJUST(CURSOR,COPYLENGTH);
1061  15    4:6    46                GETLEADING;
1062  15    4:6    49                CURSOR:=MAX(CURSOR,STUFFSTART);
1063  15    4:6    58                CENTERCURSOR(TRASH,MIDDLE,TRUE)
1064  15    4:5    65              END;
1065  15    4:2    68      END (* CH='B' *)
1066  15    4:1    68    ELSE
1067  15    4:2    70      IF CH='F' THEN EXIT(EDITCORE);
1068  15    4:1    79    SHOWCURSOR;
1069  15    4:1    82    NEXTCOMMAND;
1070  15    4:0    84  END(*COPY*);
1071  15    4:0    02
1072  15    5:0     1  PROCEDURE DUMP;
1073  15    5:0     0  BEGIN
1074  15    5:1     0    NEXTCOMMAND;
1075  15    5:0     2  END(* DUMP *);
1076  15    5:0    14
1077  15    6:0     1  PROCEDURE FIND; FORWARD;
1078  15    6:0     1
1079  15    7:0     1  PROCEDURE INSERTIT; FORWARD;
1080  15    7:0     1
1081  15    8:0     1  PROCEDURE JUMP;
1082  15    8:0     1  VAR CH: CHAR;
1083  15    8:0     2
1084  15    9:0     1  PROCEDURE JUMPMARKER;
1085  15    9:0     1  VAR
1086  15    9:0     1    I: INTEGER;
1087  15    9:0     2    MNAME: PACKED ARRAY [0..7] OF CHAR;
1088  15    9:0     0  BEGIN
1089  15    9:1     0    WITH PAGEZERO DO
1090  15    9:2     0      BEGIN
1091  15    9:3     0        GETNAME('JUMP TO',MNAME);
1092  15    9:3    15        IF MNAME<>'        ' THEN
1093  15    9:4    33          BEGIN
1094  15    9:5    33            I:=0;
1095  15    9:5    36            WHILE (I<COUNT) AND (MNAME<>NAME[I]) DO I:=I+1;
1096  15    9:5    62            IF MNAME<>NAME[I] THEN
1097  15    9:6    75              ERROR('NOT THERE.',NONFATAL)
1098  15    9:5    89            ELSE
1099  15    9:6    94              BEGIN
1100  15    9:7    94                CURSOR:=POFFSET[I];
```

```
1101  15     9:7     03                GETLEADING;
1102  15     9:7     06                CURSOR:=MAX(CURSOR,STUFFSTART);
1103  15     9:7     15                CENTERCURSOR(TRASH,MIDDLE,FALSE)
1104  15     9:6     22              END;
1105  15     9:4     25            END;
1106  15     9:2     25        END;
1107  15     9:0     25 END; (* JUMPMARKER *)
1108  15     9:0     40
1109  15     8:0      0 BEGIN (* JUMP *)
1110  15     8:1      0   PROMPTLINE:=' JUMP: B(EGINNING E(ND M(ARKER <ESC>';
1111  15     8:1     44   PROMPT;
1112  15     8:1     47   NEEDPROMPT:=TRUE;  (* NEED TO REDISPLAY EDIT: PROMPTLINE! *)
1113  15     8:1     51   REPEAT
1114  15     8:2     51     CH:=UCLC(GETCH);
1115  15     8:2     63     IF CH='B' THEN
1116  15     8:3     68       BEGIN
1117  15     8:4     68         CURSOR:=1;
1118  15     8:4     71         GETLEADING;
1119  15     8:4     74         CURSOR:=STUFFSTART;
1120  15     8:4     77         CENTERCURSOR(TRASH,1,FALSE)
1121  15     8:3     82       END
1122  15     8:2     85     ELSE
1123  15     8:3     87       IF CH='E' THEN
1124  15     8:4     92         BEGIN
1125  15     8:5     92           CURSOR:=BUFCOUNT-1;
1126  15     8:5     97           CENTERCURSOR(TRASH,SCREENHEIGHT-1,FALSE);
1127  15     8:4     07         END
1128  15     8:3     07       ELSE
1129  15     8:4     09         IF CH='M' THEN JUMPMARKER
1130  15     8:4     14         ELSE IF CH<>CHR(ESC) THEN ERRWAIT;
1131  15     8:1     28   UNTIL (CH IN ['B','E','M',CHR(ESC)]);
1132  15     8:1     51   NEXTCOMMAND;
1133  15     8:0     53 END;
1134  15     8:0     68
1135  15    10:D      1 PROCEDURE DEFMACRO;
1136  15    10:0      0 BEGIN
1137  15    10:1      0   WITH PAGEZERO DO IF FILLING AND NOT AUTOINDENT THEN
1138  15    10:3     10     BEGIN
1139  15    10:4     10       BLANKCRT(1);
1140  15    10:4     14       THEFIXER(CURSOR,REPEATFACTOR,TRUE);
1141  15    10:4     20       CENTERCURSOR(TRASH,MIDDLE,TRUE);
```

```
1142  15   10:3   30      END
1143  15   10:2   30    ELSE ERROR('INAPPROPRIATE ENVIRONMENT',NONFATAL);
1144  15   10:1   64    COPYOK:=FALSE;
1145  15   10:1   68    SHOWCURSOR;
1146  15   10:1   71    NEXTCOMMAND;
1147  15   10:0   73  END;
1148  15   10:0   86
1149  15   11:D    1  PROCEDURE SETMARKER;
1150  15   11:D    1  LABEL 1;
1151  15   11:D    1  VAR
1152  15   11:D    1    I,SLOT: INTEGER;
1153  15   11:D    3    MNAME: PACKED ARRAY [0..7] OF CHAR;
1154  15   11:0    0  BEGIN
1155  15   11:1    0    WITH PAGEZERO DO
1156  15   11:2    0      BEGIN
1157  15   11:3    0        NEEDPROMPT:=TRUE;
1158  15   11:3    4        COUNT:=MIN(10,COUNT);
1159  15   11:3   16        IF COUNT=10 THEN
1160  15   11:4   23          BEGIN
1161  15   11:5   23            BLANKCRT(1);
1162  15   11:5   27            FOR I:=0 TO COUNT-1 DO
1163  15   11:6   42              WRITELN(I,') ',NAME[I]);
1164  15   11:5   89            MSG:='MARKER OVFLW.  WHICH ONE TO REPLACE?';
1165  15   11:5   33            PUTMSG; CH:=GETCH;
1166  15   11:5   43            CENTERCURSOR(TRASH,MIDDLE,TRUE);
1167  15   11:5   53            IF NOT (CH IN ['0'..'9']) THEN GOTO 1;
1168  15   11:5   71            SLOT:=ORD(CH)-ORD('0')
1169  15   11:4   73          END
1170  15   11:3   76        ELSE
1171  15   11:4   78          SLOT:=COUNT;
1172  15   11:3   83        GETNAME('SET',MNAME);
1173  15   11:3   94        IF MNAME<>'        ' THEN
1174  15   11:4   12          BEGIN
1175  15   11:5   12            FOR I:=0 TO COUNT-1 DO
1176  15   11:6   27              IF NAME[I]=MNAME THEN SLOT:=I;
1177  15   11:5   50            NAME[SLOT]:=MNAME;
1178  15   11:5   60            POFFSET[SLOT]:=CURSOR;
1179  15   11:5   68            IF SLOT=COUNT THEN COUNT:=COUNT+1
1180  15   11:4   78          END;
1181  15   11:2   83      END;
1182  15   11:1   83 1:END;
```

```
1183  15   11:1    02
1184  15   12:D     1 PROCEDURE SETSTUFF;
1185  15   12:D     1 VAR CH: CHAR;
1186  15   12:0     0 BEGIN
1187  15   12:1     0   PROMPTLINE:=' SET: E(NVIRONMENT M(ARKER <ESC>';
1188  15   12:1    40   PROMPT; NEEDPROMPT:=TRUE;
1189  15   12:1    47   REPEAT
1190  15   12:2    47     CH:=UCLC(GETCH);
1191  15   12:2    59     IF CH='E' THEN EXIT(EDITCORE)
1192  15   12:2    68     ELSE
1193  15   12:3    70       IF CH='M' THEN SETMARKER
1194  15   12:3    75       ELSE IF CH<>CHR(ESC) THEN ERRWAIT;
1195  15   12:1    89   UNTIL CH IN ['E','M',CHR(ESC)];
1196  15   12:1    11   SHOWCURSOR;
1197  15   12:1    14   NEXTCOMMAND;
1198  15   12:0    16 END(* SETSTUFF *);
1199  15   12:0    30
1200  15   13:D     1 PROCEDURE VERIFY;
1201  15   13:0     0 BEGIN
1202  15   13:1     0   CENTERCURSOR(TRASH,MIDDLE,TRUE);
1203  15   13:1    10   SHOWCURSOR;
1204  15   13:1    13   NEXTCOMMAND
1205  15   13:0    13 END (* VERIFY *);
1206  15   13:0    28
1207  15   14:D     1 PROCEDURE XMACRO;
1208  15   14:D     1 VAR
1209  15   14:D     1   SAVEC,I: INTEGER;
1210  15   14:D     3   SAVE:PACKED ARRAY [0..MAXSTRING] OF CHAR;
1211  15   14:0     0 BEGIN
1212  15   14:1     0   PROMPTLINE:=' EXCHANGE: TEXT [<BS> A CHAR] [<ESC> ESCAPES; <ETX> ACCEPTS]';
1213  15   14:1    68   PROMPT; NEEDPROMPT:=TRUE;
1214  15   14:1    75   SHOWCURSOR;
1215  15   14:1    78   SAVEC:=CURSOR;
1216  15   14:1    81   I:=0;
1217  15   14:1    84   REPEAT
1218  15   14:2    84     CH:=GETCH;
1219  15   14:2    91     IF MAPTOCOMMAND(CH)=LEFT THEN
1220  15   14:3    01       BEGIN
1221  15   14:4    01         IF (CURSOR>SAVEC) THEN
1222  15   14:5    06           BEGIN
1223  15   14:6    06             I:=I-1; CURSOR:=CURSOR-1; (* DECREMENT BOTH PTRS *)
```

```
1224  15   14:6   16              EBUF^[CURSOR]:=SAVE[I]; (* RESTORE BUFFER *)
1225  15   14:6   24              WRITE(CHR(BS),EBUF^[CURSOR],CHR(BS));
1226  15   14:5   55            END
1227  15   14:3   55        END
1228  15   14:2   55      ELSE
1229  15   14:3   57        IF CH=CHR(EOL) THEN BEGIN ERRWAIT; SHOWCURSOR END
1230  15   14:3   68        ELSE
1231  15   14:4   70          IF NOT (CH IN [CHR(ETX),CHR(ESC)]) AND (EBUF^[CURSOR]<>CHR(EOL)) THEN
1232  15   14:5   91            BEGIN
1233  15   14:6   91              IF NOT (CH IN [' '..'^']) THEN CH:='?';
1234  15   14:6   18              SAVE[I]:=EBUF^[CURSOR];
1235  15   14:6   26              EBUF^[CURSOR]:=CH;
1236  15   14:6   31              I:=I+1; CURSOR:=CURSOR+1;
1237  15   14:6   41              WRITE(CH)
1238  15   14:5   49            END;
1239  15   14:1   49    UNTIL CH IN [CHR(ETX),CHR(ESC)];
1240  15   14:1   62    IF CH=CHR(ESC) THEN
1241  15   14:2   69      BEGIN
1242  15   14:3   69        CURSOR:=SAVEC;
1243  15   14:3   72        MOVELEFT(SAVE[0],EBUF^[CURSOR],I);
1244  15   14:3   81        SHOWCURSOR; WRITE(SAVE:I); SHOWCURSOR
1245  15   14:2   96      END;
1246  15   14:1   99    NEXTCOMMAND;
1247  15   14:0   01 END (* XMACRO *);
1248  15   14:0   16
1249  15   15:D    1 PROCEDURE ZAPIT;
1250  15   15:0    0 BEGIN
1251  15   15:1    0   IF ABS(LASTPAT-CURSOR)>80 THEN
1252  15   15:2    8     BEGIN
1253  15   15:3    8       PROMPTLINE:=
1254  15   15:3   11 ' WARNING! YOU ARE ABOUT TO ZAP MORE THAN 80 CHARS, DO YOU WISH TO ZAP? (Y/N)';
1255  15   15:3   92       PROMPT;
1256  15   15:3   95       NEEDPROMPT:=TRUE;
1257  15   15:3   99       IF UCLC(GETCH)<>'Y' THEN
1258  15   15:4   13         BEGIN
1259  15   15:5   13           SHOWCURSOR;
1260  15   15:5   16           NEXTCOMMAND;
1261  15   15:5   18           EXIT(ZAPIT)
1262  15   15:4   22         END;
1263  15   15:2   22     END;
1264  15   15:1   22   IF OKTODEL(MIN(CURSOR,LASTPAT),MAX(CURSOR,LASTPAT)) THEN
```

```
1265  15   15:2    43      BEGIN
1266  15   15:3    43        COPYLINE:=FALSE;
1267  15   15:3    47        READJUST(MIN(CURSOR,LASTPAT),-ABS(CURSOR-LASTPAT));
1268  15   15:3    62        IF CURSOR>LASTPAT THEN
1269  15   15:4    67          MOVELEFT(EBUF^[CURSOR],EBUF^[LASTPAT],BUFCOUNT-CURSOR)
1270  15   15:3    78        ELSE
1271  15   15:4    80          MOVELEFT(EBUF^[LASTPAT],EBUF^[CURSOR],BUFCOUNT-LASTPAT);
1272  15   15:3    91        BUFCOUNT:=BUFCOUNT-ABS(CURSOR-LASTPAT);
1273  15   15:3    99        CURSOR:=LASTPAT;
1274  15   15:3    02        CENTERCURSOR(TRASH,MIDDLE,TRUE);
1275  15   15:2    12      END;
1276  15   15:1    12    SHOWCURSOR;
1277  15   15:1    15    NEXTCOMMAND;
1278  15   15:0    17  END;
1279  15   15:0    30
1280  15   15:0    30  (*$I COMMAND   *)
1280  15   15:0    30  (*$I INSERTIT  *)
1281  15    7:D     1  PROCEDURE INSERTIT;
1282  15    7:D     1  CONST
1283  15    7:D     1    FUDGEFACTOR=10;
1284  15    7:D     1  VAR
1285  15    7:D     1    THEREST,LEFTPART,SAVEBUFCOUNT: PTRTYPE;
1286  15    7:D     4    CLEARED,WARNED,OK,NOTEXTYET,EXITPROMPT,FIRSTLINE: BOOLEAN;
1287  15    7:D    10    SPACES,LMOVE,X,LINE,EOLDIST,RJUST: INTEGER;
1288  15    7:D    16    CONTEXT: PACKED ARRAY [0..MAXSTRING] OF CHAR;
1289  15    7:D    80
1290  15   16:D     1  PROCEDURE SLAMRIGHT;
1291  15   16:D     1  (* MOVE (SLAM) THE PORTION OF THE EBUF^ TO THE RIGHT OF (AND INCLUDING)
1292  15   16:D     1     THE CURSOR SO THAT THE LAST NUL IN THE FILE (EBUF^[BUFCOUNT]) IS NOW AT
1293  15   16:D     1     EBUF^[BUFSIZE].  THEREST POINTS TO THE BEGINNING OF THE RIGHT-JUSTIFIED
1294  15   16:D     1     TEXT. *)
1295  15   16:0     0  BEGIN
1296  15   16:1     0    GETLEADING;
1297  15   16:1     3    THEREST:=BUFSIZE-(BUFCOUNT-CURSOR);
1298  15   16:1    11    LMOVE:=BUFCOUNT-CURSOR+1;
1299  15   16:1    19    MOVERIGHT(EBUF^[CURSOR],EBUF^[THEREST],LMOVE);
1300  15   16:1    32    GETLEADING; (* SET BLANKS *)
1301  15   16:1    35    IF THEREST-CURSOR<MAXSTRING THEN
1302  15   16:2    44      BEGIN
1303  15   16:3    44        ERROR('NO ROOM TO INSERT.',NONFATAL);
1304  15   16:3    69        SHOWCURSOR;
```

```
1305  15   16:3    72        NEXTCOMMAND;
1306  15   16:3    74        EXIT(INSERTIT)
1307  15   16:2    78      END;
1308  15   16:2    78    (* OPTIONAL INDENTATION *)
1309  15   16:1    78    EBUF^[THEREST-2]:=CHR(DLE); EBUF^[THEREST-1]:=CHR(BLANKS+32);
1310  15   16:0    98 END;
1311  15   16:0    10
1312  15   17:D     1 PROCEDURE WRAPUP;
1313  15   17:D     1 (* GIVEN THE NEW VALUE OF THE CURSOR (ONE PAST THE LAST VALID CHARACTER
1314  15   17:D     1    INSERTED INTO THE BUFFER), PUT BACK TOGETHER THE TWO HALVES OF THE
1315  15   17:D     1    BUFFER.  THEN, TO POLISH IT OFF, UPDATE THE SCREEN SO THAT THE REST OF
1316  15   17:D     1    THE EDITOR CAN COPE *)
1317  15   17:D     1 VAR PTR: PTRTYPE;
1318  15   17:D     2     LNGTH: INTEGER;
1319  15   17:0     0 BEGIN
1320  15   17:1     0   WITH PAGEZERO DO
1321  15   17:2     0     IF NOTEXTYET AND (NOT FIRSTLINE) AND
1322  15   17:2     8        ((NOT FILLING) OR AUTOINDENT) AND (CH<>CHR(ESC))
1323  15   17:2    22     THEN (* WE WANT THE BLANKS BEFORE THEREST *)
1324  15   17:3    25       BEGIN
1325  15   17:4    25         BUFCOUNT:=BUFCOUNT+2;
1326  15   17:4    30         THEREST:=THEREST-2; LMOVE:=LMOVE+2;
1327  15   17:4    46         CURSOR:=SCAN(-MAXCHAR,=CHR(EOL),EBUF^[CURSOR-1])+CURSOR;
1328  15   17:3    64       END;
1329  15   17:1    64   MOVELEFT(EBUF^[THEREST],EBUF^[CURSOR],LMOVE);
1330  15   17:1    77   READJUST(LEFTPART+1,CURSOR-(LEFTPART+1));
1331  15   17:1    92   BUFCOUNT:=BUFCOUNT+CURSOR-(LEFTPART+1);
1332  15   17:1    03   WITH PAGEZERO DO
1333  15   17:2    03     IF FILLING AND NOT AUTOINDENT AND (CH=CHR(ETX)) THEN
1334  15   17:3    19       BEGIN THEFIXER(CURSOR,1,FALSE); FIRSTLINE:=FALSE; FINDXY(X,LINE) END;
1335  15   17:1    38   UPSCREEN(FIRSTLINE,EXITPROMPT OR (CH=CHR(ESC)),LINE);
1336  15   17:1    56   GETLEADING;
1337  15   17:1    59   CURSOR:=MAX(CURSOR,STUFFSTART);
1338  15   17:1    68   LASTPAT:=LEFTPART+1;
1339  15   17:1    75   COPYOK:=TRUE; COPYSTART:=LASTPAT; COPYLENGTH:=CURSOR-LASTPAT;
1340  15   17:1    89   NEXTCOMMAND
1341  15   17:0    89 END;
1342  15   17:0    04
1343  15   18:D     3 FUNCTION CHECK(VALUE:INTEGER): BOOLEAN;
1344  15   18:D     4   (* VALUE IS THE POTENTIAL VALUE OF THE CURSOR.  IF IT IS NOT IN LEGAL
1345  15   18:D     4      RANGE THEN CHECK IS FALSE.  THIS FUNCTION ALSO WARNS THE USER IF
```

```
1346  15    18:0      4      S/HE IS GETTING TOO CLOSE TO OVERFLOWING THE BUFFER *)
1347  15    18:0      0 BEGIN
1348  15    18:1      0   CHECK:=TRUE;
1349  15    18:1      3   IF VALUE<=LEFTPART THEN
1350  15    18:2     10     BEGIN
1351  15    18:3     10       OK:=FALSE;  CHECK:=FALSE;
1352  15    18:3     17       ERROR('NO INSERTION TO BACK OVER.',NONFATAL); PROMPT;
1353  15    18:3     53       GOTOXY(X,LINE)
1354  15    18:2     62     END
1355  15    18:1     62   ELSE
1356  15    18:2     64     IF VALUE>=THEREST-MAXCHAR THEN
1357  15    18:3     75       BEGIN
1358  15    18:4     75         IF NOT WARNED THEN
1359  15    18:5     81           BEGIN
1360  15    18:6     81             ERROR('PLEASE FINISH UP THE INSERTION',NONFATAL); PROMPT;
1361  15    18:6     21             GOTOXY(X,LINE);
1362  15    18:6     30             WARNED:=TRUE
1363  15    18:5     30           END;
1364  15    18:4     34         IF VALUE>THEREST-FUDGEFACTOR THEN
1365  15    18:5     43           BEGIN
1366  15    18:6     43             ERROR('BUFFER OVERFLOW!!!!',NONFATAL);
1367  15    18:6     69             WRAPUP;
1368  15    18:6     71             EXIT(INSERTIT);
1369  15    18:5     75           END
1370  15    18:3     75       END
1371  15    18:0     75 END;
1372  15    18:0     88
1373  15    19:0      1 PROCEDURE SPACEOVER;
1374  15    19:0      1 (* THIS PROCEDURE HANDLES SPACES AND TABS INSERTED INTO THE BUFFER *)
1375  15    19:0      0 BEGIN
1376  15    19:1      0   IF CH=CHR(HT) THEN SPACES:=8-X+ORD(ODD(X) AND ODD(248)) ELSE SPACES:=1;
1377  15    19:1     27   IF CHECK(CURSOR+SPACES) THEN
1378  15    19:2     38     BEGIN
1379  15    19:3     38       FILLCHAR(EBUF^[CURSOR],SPACES,' ');
1380  15    19:3     47       CURSOR:=CURSOR+SPACES
1381  15    19:2     48     END
1382  15    19:0     54 END;
1383  15    19:0     66
1384  15    20:0      1 PROCEDURE FIXUP; FORWARD;
1385  15    20:0      1
1386  15    21:0      1 PROCEDURE ENDLINE;
```

```
1387  15   21:0    1 (* FIRST, IF THERE WAS NO TEXT INSERTED ON THE CURRENT LINE, THEN CONVERT
1388  15   21:0    1    ALL OF THE SPACES TO BLANK COMPRESSION CODES.  THEN INSERT AN <EOL> INTO
1389  15   21:0    1    THE BUFFER FOLLOWED BY THE APPROPRIATE NUMBER OF SPACES FOR THE
1390  15   21:0    1    INDENTATION. *)
1391  15   21:0    0 BEGIN
1392  15   21:1    0   WITH PAGEZERO DO
1393  15   21:2    0     BEGIN
1394  15   21:3    0       IF NOTEXTYET THEN FIXUP;
1395  15   21:3    7       EBUF^[CURSOR]:=CHR(EOL);
1396  15   21:3   12       IF AUTOINDENT THEN GETLEADING
1397  15   21:3   17       ELSE
1398  15   21:4   22         IF FILLING THEN
1399  15   21:5   27           BEGIN
1400  15   21:6   27             GETLEADING;
1401  15   21:6   30             IF EBUF^[STUFFSTART]=CHR(EOL) THEN (* EMPTY LINE *)
1402  15   21:7   38               BLANKS:=PARAMARGIN
1403  15   21:6   38             ELSE BLANKS:=LMARGIN
1404  15   21:5   45           END
1405  15   21:4   50         ELSE BLANKS:=0;
1406  15   21:3   55       IF CHECK(CURSOR+BLANKS+1) THEN
1407  15   21:4   66         BEGIN
1408  15   21:5   66           FILLCHAR(EBUF^[CURSOR+1],BLANKS,' ');
1409  15   21:5   75           CURSOR:=CURSOR+BLANKS+1
1410  15   21:4   78         END;
1411  15   21:3   82       NOTEXTYET:=TRUE;
1412  15   21:2   86     END;
1413  15   21:0   86 END;
1414  15   21:0   98
1415  15   22:0    1 PROCEDURE BACKUP;
1416  15   22:0    1 (* IF THE CH IS A BACKSPACE THEN DECREMENT CURSOR BY 1.  IF THIS WOULD
1417  15   22:0    1    RESULT IN BACKING OVER AN <EOL> OR A BLANK COMPRESSION CODE, THEN FALL
1418  15   22:0    1    INTO THE CODE FOR A <DEL> (ALSO CHANGING THE CH TO <DEL> FOR COMMUNICATION
1419  15   22:0    1    TO THE OUTER BLOCK)  *)
1420  15   22:0    1 VAR PTR: PTRTYPE;
1421  15   22:0    0 BEGIN
1422  15   22:1    0   IF CH=CHR(DC1) THEN
1423  15   22:2    5     BEGIN GETLEADING; IF CHECK(LINESTART) THEN CURSOR:=LINESTART END
1424  15   22:1   18   ELSE
1425  15   22:2   20     IF (CH=CHR(BS)) AND
1426  15   22:2   25        NOT( (EBUF^[CURSOR-2]=CHR(DLE)) OR (EBUF^[CURSOR-1]=CHR(EOL)) ) THEN
1427  15   22:3   46       BEGIN
```

```
1428  15    22:4     46          IF CURSOR<LEFTPART+2 THEN OK:=FALSE ELSE CURSOR:=CURSOR-1;
1429  15    22:3     66        END
1430  15    22:2     66      ELSE
1431  15    22:3     68        BEGIN (* A <DEL> OR EQUIVALENT *)
1432  15    22:4     68          CH:=CHR(DEL); (* TELL THE CRT DRIVER THAT THE LINE HAS CHANGED *)
1433  15    22:4     73          GETLEADING;
1434  15    22:4     76          IF CHECK(LINESTART-1) THEN CURSOR:=LINESTART-1;
1435  15    22:4     90          NOTEXTYET:=FALSE; (* THANK YOU SHAWN! *)
1436  15    22:3     94        END
1437  15    22:0     94 END;
1438  15    22:0     06
1439  15    20:D      1 PROCEDURE FIXUP;
1440  15    20:D      1 (* CONVERT THE INDENTATION SPACES INTO BLANK COMPRESSION CODES, AND MOVE
1441  15    20:D      1    THE CURRENT LINE AROUND ACCORDINGLY *)
1442  15    20:0      0 BEGIN
1443  15    20:0      0   (* FIRST COMPRESS THE CURRENT LINE *)
1444  15    20:1      0   EBUF^[CURSOR]:=CHR(EOL); (* FOOL GETLEADING *)
1445  15    20:1      5   GETLEADING;
1446  15    20:1      8   IF BYTES >= 2 THEN (* OK TO PUT IN <DLE> # AS IT STANDS *)
1447  15    20:2     13     MOVELEFT(EBUF^[STUFFSTART],EBUF^[LINESTART+2],CURSOR-STUFFSTART)
1448  15    20:1     26   ELSE
1449  15    20:2     28     IF CHECK(CURSOR+2-BYTES) THEN
1450  15    20:3     39       MOVERIGHT(EBUF^[STUFFSTART],EBUF^[STUFFSTART+2-BYTES],CURSOR-STUFFSTART)
1451  15    20:2     54     ELSE BEGIN OK:=FALSE; EXIT(FIXUP) END;
1452  15    20:1     64   CURSOR:=CURSOR-(BYTES-2);
1453  15    20:1     71   EBUF^[LINESTART]:=CHR(DLE); EBUF^[LINESTART+1]:=CHR(32+BLANKS);
1454  15    20:0     85 END;
1455  15    20:0     98
1456  15    23:D      1 PROCEDURE INSERTCH;
1457  15    23:D      1   (* THIS PROCEDURE INSERTS A SINGLE CHARACTER INTO THE BUFFER. IT ALSO
1458  15    23:D      1      HANDLES ALL OF THE CONTROL CODES (EOL,BS,DEL) AND BUFFER OVER- AND
1459  15    23:D      1      UNDER- FLOW CONDITIONS.  INSERTCH IS CALLED BY THE CRT HANDLER *)
1460  15    23:0      0 BEGIN
1461  15    23:1      0   REPEAT
1462  15    23:2      0   OK:=TRUE; (* NO ERRORS THAT INVALIDATE THE CURRENT CHARACTER HAVE OCCURED *)
1463  15    23:2      4   CH:=GETCH;
1464  15    23:2     11   IF MAPTOCOMMAND(CH)=LEFT THEN CH:=CHR(BS);
1465  15    23:2     26   IF ORD(CH) IN [SP,HT,EOL,BS,DEL,ETX,ESC,DC1] THEN
1466  15    23:3     59     BEGIN
1467  15    23:3     59       (* <ETX> AND <ESC> ARE HANDLED IN THE BODY OF INSERTIT *)
1468  15    23:4     59       IF ORD(CH) IN [SP,HT] THEN SPACEOVER
```

```
1469  15   23:4   72        ELSE
1470  15   23:5   76          IF ORD(CH)=EOL THEN ENDLINE
1471  15   23:5   81          ELSE
1472  15   23:6   85            IF ORD(CH) IN [DC1,BS,DEL] THEN BACKUP;
1473  15   23:3   09      END
1474  15   23:2   09    ELSE
1475  15   23:3   11      BEGIN (* A CHARACTER TO INSERT! *)
1476  15   23:4   11        IF (CH<'!') OR (CH>'^') THEN CH:='?'; (* NO NON-PRINTING CHARACTERS *)
1477  15   23:4   23        IF NOTEXTYET THEN FIXUP;
1478  15   23:4   30        IF CHECK(CURSOR+1) AND OK THEN
1479  15   23:5   43          BEGIN
1480  15   23:6   43            NOTEXTYET:=FALSE;
1481  15   23:6   47            EBUF^[CURSOR]:=CH;
1482  15   23:6   52            CURSOR:=CURSOR+1
1483  15   23:5   53          END;
1484  15   23:3   57      END;
1485  15   23:1   57  UNTIL OK;
1486  15   23:0   62 END;
1487  15   23:0   76
1488  15   24:D    1 PROCEDURE POPDOWN;
1489  15   24:D    1 (* DISPLAYS CONTEXT, DOING AN IMPLIED SCROLLUP IF NEC. *)
1490  15   24:0    0 BEGIN
1491  15   24:1    0   IF CLEARED THEN ERASETOEOL(X,LINE)
1492  15   24:1   11   ELSE BEGIN CLEARED:=TRUE; ERASEOS(X,LINE) END;
1493  15   24:1   29   GOTOXY(RJUST,LINE);
1494  15   24:1   38   ERASETOEOL(RJUST,LINE);
1495  15   24:1   47   WRITE(CHR(LF));
1496  15   24:1   55   IF LINE=SCREENHEIGHT THEN BEGIN EXITPROMPT:=TRUE; LINE:=SCREENHEIGHT-1 END;
1497  15   24:1   72   WRITE(CONTEXT:EOLDIST);
1498  15   24:1   87   FIRSTLINE:=FALSE; (* SAYS THAT THE WHOLE SCREEN HAS BEEN AFFECTED. *)
1499  15   24:0   91 END;
1500  15   24:0   04
1501  15   25:D    1 PROCEDURE WRITESP(CH:CHAR;HOWMANY:INTEGER);
1502  15   25:0    0 BEGIN
1503  15   25:1    0   IF X+HOWMANY<=SCREENWIDTH THEN WRITE(CH:HOWMANY);
1504  15   25:1   17   IF X+HOWMANY>=SCREENWIDTH THEN
1505  15   25:2   26     BEGIN
1506  15   25:3   26       GOTOXY(SCREENWIDTH,LINE);
1507  15   25:3   33       IF X+HOWMANY>SCREENWIDTH THEN
1508  15   25:4   42         BEGIN WRITE('!'); GOTOXY(SCREENWIDTH,LINE) END
1509  15   25:2   57     END;
```

```
1510  15    25:1     57    X:=MIN(SCREENWIDTH,X+HOWMANY)
1511  15    25:0     63 END;
1512  15    25:0     84
1513  15    26:D      1 PROCEDURE CLEANSCREEN;
1514  15    26:D      1 (* CODE TO, IF POSSIBLE, ONLY ERASE THE LINE, OTHERWISE CLEAR
1515  15    26:D      1    THE SCREEN.  THEN CALL POPDOWN *)
1516  15    26:0      0 BEGIN
1517  15    26:1      0   FIRSTLINE:=FALSE;
1518  15    26:1      4   IF CLEARED THEN
1519  15    26:2      9     BEGIN
1520  15    26:3      9       IF X<SCREENWIDTH THEN ERASETOEOL(X,LINE)
1521  15    26:2     22     END
1522  15    26:1     25   ELSE
1523  15    26:2     27     BEGIN
1524  15    26:3     27       CLEARED:=TRUE; ERASEOS(X,LINE);
1525  15    26:2     40     END;
1526  15    26:1     40   LINE:=LINE+1;
1527  15    26:1     48   IF LINE>SCREENHEIGHT THEN
1528  15    26:2     55     BEGIN
1529  15    26:3     55       LINE:=LINE-1;
1530  15    26:3     63       WRITELN;
1531  15    26:3     69       EXITPROMPT:=TRUE
1532  15    26:2     69     END;
1533  15    26:1     73   IF EOLDIST<>0 THEN POPDOWN
1534  15    26:0     80 END;
1535  15    26:0     94
1536  15    27:D      1 PROCEDURE POPOV;
1537  15    27:D      1 (* WHEN IN FILLING MODE, THIS PROCEDURE IS CALLED WHEN A LINE IS OVERFLOWED
1538  15    27:D      1    (X >= RIGHTMARGIN).  THE WORD IS SCANNED OFF AND "POPPED" DOWN TO THE
1539  15    27:D      1    NEXT LINE. *)
1540  15    27:D      1 VAR
1541  15    27:D      1   WLENGTH: INTEGER;
1542  15    27:D      2   SAVE,PTR: PTRTYPE;
1543  15    27:D      4   WORD: PACKED ARRAY [0..MAXSW] OF CHAR;
1544  15    27:0      0 BEGIN
1545  15    27:1      0   IF NOTEXTYET THEN FIXUP;
1546  15    27:1      7   PTR:=MAX(SCAN(-MAXCHAR,='-',EBUF^[CURSOR-1]),
1547  15    27:1     21            SCAN(-MAXCHAR,=' ',EBUF^[CURSOR-1]))+CURSOR;
1548  15    27:1     44   WLENGTH:=CURSOR-PTR;
1549  15    27:1     49   WITH PAGEZERO DO IF WLENGTH>=RMARGIN-LMARGIN THEN
1550  15    27:3     60     BEGIN
```

```
1551  15   27:4   60        WRITESP(CH,1);
1552  15   27:4   64        EXIT(POPOV)
1553  15   27:3   68      END;
1554  15   27:1   68    IF CH='-' THEN WRITE('-');
1555  15   27:1   81    GOTOXY(X-WLENGTH+1,LINE);
1556  15   27:1   94    ERASETOEOL(X-WLENGTH+1,LINE);
1557  15   27:1   07    MOVERIGHT(EBUF^[PTR],EBUF^[PTR+3],WLENGTH);
1558  15   27:1   18    MOVELEFT(EBUF^[PTR+3],WORD,WLENGTH);
1559  15   27:1   29    CURSOR:=CURSOR+3;
1560  15   27:1   34    EBUF^[PTR]:=CHR(EOL);
1561  15   27:1   39    EBUF^[PTR+1]:=CHR(DLE);
1562  15   27:1   46    WITH PAGEZERO DO IF AUTOINDENT THEN
1563  15   27:3   51      BEGIN
1564  15   27:4   51        SAVE:=CURSOR; (* SET BLANKS TO THE INDENTATION OF THE LINE ABOVE *)
1565  15   27:4   54        CURSOR:=PTR;
1566  15   27:4   57        GETLEADING;
1567  15   27:4   60        CURSOR:=SAVE
1568  15   27:3   60      END
1569  15   27:2   63    ELSE
1570  15   27:3   65      BLANKS:=LMARGIN;
1571  15   27:1   70    EBUF^[PTR+2]:=CHR(BLANKS+32);
1572  15   27:1   79    CLEANSCREEN;
1573  15   27:1   81    X:=BLANKS;
1574  15   27:1   85    GOTOXY(X,LINE); WRITE(WORD:WLENGTH);
1575  15   27:1   04    X:=X+WLENGTH;
1576  15   27:1   12    NOTEXTYET:=FALSE
1577  15   27:0   12  END;
1578  15   27:0   28
1579  15    7:0    0  BEGIN (* INSERT *)
1580  15    7:1    0    CLEARED:=FALSE;
1581  15    7:1    3    EOLDIST:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR]);
1582  15    7:1   16    MOVELEFT(EBUF^[CURSOR],CONTEXT[0],EOLDIST);
1583  15    7:1   25    RJUST:=SCREENWIDTH-EOLDIST;
1584  15    7:1   30    SLAMRIGHT;
1585  15    7:1   32    SAVEBUFCOUNT:=BUFCOUNT;
1586  15    7:1   35    PROMPTLINE:=INSERTPROMPT;
1587  15    7:1   43    PROMPT;
1588  15    7:1   46    EXITPROMPT:=FALSE;  NEEDPROMPT:=TRUE;
1589  15    7:1   53    LEFTPART:=CURSOR-1;
1590  15    7:1   58    NOTEXTYET:=FALSE;
1591  15    7:1   61    FINDXY(X,LINE);  GOTOXY(X,LINE);
```

```
1592  15    7:1    73   ERASETOEOL(X,LINE);
1593  15    7:1    78   FIRSTLINE:=TRUE;
1594  15    7:1    81   IF EOLDIST<>0 THEN (* A CONTEXT NEEDS TO BE DISPLAYED *)
1595  15    7:2    86     IF RJUST>X THEN (* AND IT WILL FIT ON THE CURRENT LINE ... *)
1596  15    7:3    91       BEGIN
1597  15    7:4    91         GOTOXY(RJUST,LINE); WRITE(CONTEXT:EOLDIST); GOTOXY(X,LINE)
1598  15    7:3    13       END
1599  15    7:2    13     ELSE (* AND IT WON'T FIT ON THE CURRENT LINE *)
1600  15    7:3    15       BEGIN
1601  15    7:4    15         FIRSTLINE:=FALSE;
1602  15    7:4    18         ERASEOS(X,LINE);(* CLEAR THE SCREEN *)
1603  15    7:4    23         WRITELN;
1604  15    7:4    29         IF LINE=SCREENHEIGHT THEN
1605  15    7:5    34           BEGIN LINE:=SCREENHEIGHT-1; EXITPROMPT:=TRUE END;
1606  15    7:4    42         GOTOXY(RJUST,LINE+1); WRITE(CONTEXT:EOLDIST); GOTOXY(X,LINE)
1607  15    7:3    66       END;
1608  15    7:1    66   REPEAT
1609  15    7:2    66     INSERTCH;
1610  15    7:2    68     IF NOT (ORD(CH) IN [EOL,ETX,ESC,DEL,DC1]) THEN
1611  15    7:3    96       BEGIN
1612  15    7:4    96         IF TRANSLATE[CH]=LEFT THEN
1613  15    7:5    07           BEGIN IF X<=SCREENWIDTH THEN WRITE(CHR(BS),' ',CHR(BS)); X:=X-1 END
1614  15    7:4    45         ELSE
1615  15    7:5    47           IF CH=CHR(HT) THEN WRITESP(' ',SPACES)
1616  15    7:5    54           ELSE
1617  15    7:6    58             IF PAGEZERO.FILLING AND (X+1>=PAGEZERO.RMARGIN) THEN POPOV
1618  15    7:6    71             ELSE WRITESP(CH,1);
1619  15    7:4    79         IF NOT PAGEZERO.FILLING AND (X=SCREENWIDTH-8) AND (CH<>CHR(BS))
1620  15    7:4    94           THEN WRITE(CHR(BELL));
1621  15    7:4    05         IF (EOLDIST<>0) AND
1622  15    7:4    08            (X>=RJUST) AND FIRSTLINE THEN  (*RAN INTO CONTEXT *)
1623  15    7:5    16           BEGIN
1624  15    7:6    16             POPDOWN;
1625  15    7:6    18             GOTOXY(X,LINE)
1626  15    7:5    23           END;
1627  15    7:3    23       END
1628  15    7:2    23     ELSE (* CH IN [EOL,ETX,ESC,DEL,DC1] *)
1629  15    7:3    25       BEGIN
1630  15    7:4    25         IF CH=CHR(EOL) THEN
1631  15    7:5    30           BEGIN
1632  15    7:6    30             CLEANSCREEN;
```

```
1633  15    7:6    32              X:=BLANKS;
1634  15    7:6    35              GOTOXY(X,LINE);
1635  15    7:5    40            END
1636  15    7:4    40          ELSE
1637  15    7:5    42            IF CH=CHR(DEL) THEN
1638  15    7:6    49              BEGIN
1639  15    7:7    49                IF LINE<=1 THEN  (* RUBBED OUT ALL OF WHAT WAS ON THE SCREEN *)
1640  15    7:8    54                  BEGIN
1641  15    7:9    54                    BUFCOUNT:=CURSOR+1;
1642  15    7:9    59                    EBUF^[CURSOR]:=CHR(EOL);
1643  15    7:9    64                    CENTERCURSOR(LINE,MIDDLE,TRUE);
1644  15    7:9    73                    IF EOLDIST<>0 THEN POPDOWN;
1645  15    7:9    80                    IF EXITPROMPT THEN BEGIN PROMPT; EXITPROMPT:=FALSE END
1646  15    7:8    89                  END
1647  15    7:7    89                ELSE
1648  15    7:8    91                  BEGIN GOTOXY(0,LINE); CLEARED:=FALSE;
1649  15    7:9    99                        ERASETOEOL(0,LINE); LINE:=LINE-1 END;
1650  15    7:7    09                GETLEADING;
1651  15    7:7    12                X:=BLANKS-BYTES+CURSOR-LINESTART;
1652  15    7:7    21                GOTOXY(X,LINE)
1653  15    7:6    26              END
1654  15    7:5    26            ELSE
1655  15    7:6    28              IF CH=CHR(DC1) THEN
1656  15    7:7    33                BEGIN
1657  15    7:8    33                  X:=0; GOTOXY(X,LINE); ERASETOEOL(X,LINE)
1658  15    7:7    43                END;
1659  15    7:3    46        END;
1660  15    7:1    46    UNTIL CH IN [CHR(ETX),CHR(ESC)];
1661  15    7:1    59    IF CH=CHR(ESC) THEN CURSOR:=LEFTPART+1;
1662  15    7:1    71    BUFCOUNT:=SAVEBUFCOUNT;
1663  15    7:1    74    WRAPUP;
1664  15    7:0    76  END;
1665  15    7:0    92
1666  15    7:0    92
1667  15    7:0    92
1668  15    7:0    92  (*$I INSERTIT  *)
1668  15    7:0    92  (*$I MOVEIT    *)
1669  15   28:D     1  PROCEDURE MOVEIT;
1670  15   28:D     1  VAR
1671  15   28:D     1    SCROLLMARK,X,LINE,I: INTEGER;
1672  15   28:D     5    EXITPROMPT: BOOLEAN; (* PROMPT AFTER LEAVING MOVEIT! *)
```

```
1673  15   28:D    6   OLDLINE,OLDX: INTEGER;
1674  15   28:D    8   NEWDIST,DIST: INTEGER;
1675  15   28:D   10   DOFFSCREEN,ATEND,INREPLACE,INDELETE: BOOLEAN;
1676  15   28:D   14   PTR,ANCHOR,OLDCURSOR: PTRTYPE;
1677  15   28:D   17
1678  15   29:D    1 PROCEDURE SCROLLUP(BOTTOMLINE:PTRTYPE; HOWMANY: INTEGER);
1679  15   29:D    3 (* BOTTOMLINE IS THE "LINESTART" OF THE LINE TO BE SCROLLED UP *)
1680  15   29:D    3 VAR
1681  15   29:D    3   PTR: PTRTYPE;
1682  15   29:D    4   I: INTEGER;
1683  15   29:0    0 BEGIN
1684  15   29:0    0   (* DISPLAY THE NEXT LINE ON THE BOTTOM OF THE SCREEN *)
1685  15   29:1    0   I:=0;
1686  15   29:1    3   PTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[LINE1PTR])+LINE1PTR+1;
1687  15   29:1   24   WHILE (I<HOWMANY) AND (PTR<BUFCOUNT) DO
1688  15   29:2   33     BEGIN
1689  15   29:3   33       LINE1PTR:=PTR; PTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR])+PTR+1;
1690  15   29:3   54       I:=I+1
1691  15   29:2   55     END;
1692  15   29:1   61   I:=0;
1693  15   29:1   64   GOTOXY(0,SCREENHEIGHT);
1694  15   29:1   69   REPEAT
1695  15   29:2   69     I:=I+1;
1696  15   29:2   74     BLANKS:=LEADBLANKS(BOTTOMLINE,BYTES);
1697  15   29:2   84     WRITE(CHR(LF));
1698  15   29:2   92     LINEOUT(BOTTOMLINE,BYTES,BLANKS,SCREENHEIGHT);
1699  15   29:2   00     LINE:=LINE-1;
1700  15   29:1   08   UNTIL (I>=HOWMANY) OR (BOTTOMLINE>=BUFCOUNT-1);
1701  15   29:1   19   EXITPROMPT:=TRUE;
1702  15   29:0   23 END(* SCROLLUP *);
1703  15   29:0   40
1704  15   30:D    1 PROCEDURE CLEAR(X1,Y1,X2,Y2: INTEGER); FORWARD;
1705  15   30:D    5
1706  15   31:D    1 PROCEDURE CENTER;
1707  15   31:0    0 BEGIN
1708  15   31:1    0   IF INDELETE THEN
1709  15   31:2    5     BEGIN
1710  15   31:3    5       IF LINE>=SCREENHEIGHT THEN
1711  15   31:4   12         BEGIN
1712  15   31:5   12           CENTERCURSOR(LINE,2,TRUE);
1713  15   31:5   20           IF ABS(CURSOR-ANCHOR) > ABS(DIST) THEN CLEAR(0,1,MAX(X-1,0),LINE)
```

```
1714  15   31:4    49          END
1715  15   31:3    51        ELSE
1716  15   31:4    53          BEGIN
1717  15   31:5    53            CENTERCURSOR(LINE,SCREENHEIGHT-1,TRUE);
1718  15   31:5    63            GOTOXY(X,LINE);
1719  15   31:5    72            IF ABS(CURSOR-ANCHOR) > ABS(DIST) THEN WRITE(CHR(11))
1720  15   31:4    93          END;
1721  15   31:3    93        DOFFSCREEN:=TRUE;
1722  15   31:2    97      END
1723  15   31:1    97    ELSE
1724  15   31:2    99      IF (COMMAND=PARAC) AND ((DIRECTION='<') OR (LINE MOD SCREENHEIGHT=OLDLINE))
1725  15   31:2    15        THEN CENTERCURSOR(LINE,OLDLINE,TRUE)
1726  15   31:2    25        ELSE CENTERCURSOR(LINE,MIDDLE,TRUE);
1727  15   31:1    40    IF EXITPROMPT AND (COMMAND<>QUITC) THEN
1728  15   31:2    49      BEGIN
1729  15   31:3    49        PROMPT; EXITPROMPT:=FALSE
1730  15   31:2    52      END;
1731  15   31:1    56    OLDLINE:=LINE; OLDX:=X;
1732  15   31:0    68  END;
1733  15   31:0    80
1734  15   32:D     1  PROCEDURE UPMOVE;
1735  15   32:D     1  VAR I:INTEGER;
1736  15   32:0     0  BEGIN
1737  15   32:1     0    I:=1;
1738  15   32:1     3    GETLEADING;
1739  15   32:1     6    (* FIND THE LINE FIRST *)
1740  15   32:1     6    WHILE (I<=REPEATFACTOR) AND (LINESTART>1) DO
1741  15   32:2    15      BEGIN
1742  15   32:3    15        CURSOR:=LINESTART-1; (* LAST CHAR OF LINE ABOVE *)
1743  15   32:3    20        GETLEADING;
1744  15   32:3    23        LINE:=LINE-1;  I:=I+1;
1745  15   32:2    36      END;
1746  15   32:2    38    (* IF POSSIBLE SET THE CURSOR AT THE SAME X COORD WE CAME FROM.  OTHERWISE,
1747  15   32:2    38       SET IT EITHER TO THE BEGINNING OF THE BUFFER, THE BEGINNING OF TEXT
1748  15   32:2    38       ON THAT LINE, OR THE END OF THE TEXT ON THAT LINE *)
1749  15   32:1    38    CURSOR:=
1750  15   32:1    38              MAX(1,      (* THE BEGINNING OF THE BUFFER *)
1751  15   32:1    39                  MAX(STUFFSTART,  (* THE BEGINNING OF THE TEXT *)
1752  15   32:1    40                      MIN(X-BLANKS+BYTES+LINESTART, (* SAME COL *)
1753  15   32:1    49                          SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR (* EOL *)
1754  15   32:1    60                         )
```

```
1755  15    32:1     62                    )
1756  15    32:1     67                  );
1757  15    32:1     79    IF LINE<1 THEN CENTER;
1758  15    32:0     88  END(* UPALINE *);
1759  15    32:0     02
1760  15    33:D      1  PROCEDURE DOWNMOVE;
1761  15    33:D      1  VAR
1762  15    33:D      1    I: INTEGER;
1763  15    33:D      2    NEXTEOL: PTRTYPE;
1764  15    33:0      0  BEGIN
1765  15    33:1      0    I:=1;
1766  15    33:1      3    NEXTEOL:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR;
1767  15    33:1     18    WHILE (NEXTEOL<BUFCOUNT-1) AND (I<=REPEATFACTOR) DO
1768  15    33:2     29      BEGIN
1769  15    33:3     29        CURSOR:=NEXTEOL+1;
1770  15    33:3     34        NEXTEOL:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR;
1771  15    33:3     49        IF NEXTEOL<BUFCOUNT THEN
1772  15    33:4     54          BEGIN
1773  15    33:5     54            LINE:=LINE+1;
1774  15    33:5     62            I:=I+1;
1775  15    33:5     67            IF LINE=SCREENHEIGHT+1 THEN
1776  15    33:6     76              BEGIN
1777  15    33:7     76                SCROLLMARK:=CURSOR;
1778  15    33:6     80              END;
1779  15    33:4     80          END;
1780  15    33:2     80      END;
1781  15    33:1     82    IF LINE>SCREENHEIGHT THEN
1782  15    33:2     89      IF (LINE-SCREENHEIGHT>=SCREENHEIGHT) OR (INDELETE) THEN
1783  15    33:3     02        CENTER
1784  15    33:2     02      ELSE
1785  15    33:3     06        SCROLLUP(SCROLLMARK,LINE-SCREENHEIGHT);
1786  15    33:1     16    GETLEADING;
1787  15    33:1     19    (* IF POSSIBLE SET THE CURSOR AT THE SAME X COORD WE CAME FROM.  OTHERWISE,
1788  15    33:1     19       SET IT EITHER TO THE END OF THE BUFFER, THE BEGINNING OF TEXT
1789  15    33:1     19       ON THAT LINE, OR THE END OF THE TEXT ON THAT LINE *)
1790  15    33:1     19    CURSOR:=MIN(BUFCOUNT-1,       (* END OF THE BUFFER *)
1791  15    33:1     22                MAX(STUFFSTART,     (* NOT IN THE INDENTATION *)
1792  15    33:1     23                    MIN(X-BLANKS+BYTES+LINESTART (* WHERE IT WANTS TO BE *)
1793  15    33:1     30                       ,SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR
1794  15    33:1     43                       )
1795  15    33:1     45                    )
```

```
1796  15   33:1    50                    );
1797  15   33:0    62 END(* DOWNMOVE *);
1798  15   33:0    76
1799  15   34:0     1 PROCEDURE LEFTMOVE;
1800  15   34:0     0 BEGIN
1801  15   34:1     0   GETLEADING; (* SET LINESTART AND STUFFSTART *)
1802  15   34:1     3   WHILE (STUFFSTART>CURSOR-REPEATFACTOR) AND (CURSOR>REPEATFACTOR) DO
1803  15   34:2    14     BEGIN
1804  15   34:3    14       REPEATFACTOR:=REPEATFACTOR-(CURSOR-STUFFSTART+1); (* CHARS MOVED OVER *)
1805  15   34:3    23       IF EBUF^[CURSOR]=CHR(EOL) THEN CURSOR:=CURSOR-1;
1806  15   34:3    36       CURSOR:=MAX(SCAN(-MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR,1);
1807  15   34:3    58       LINE:=LINE-1;
1808  15   34:3    66       GETLEADING; (* RESET LINESTART AND STUFFSTART *)
1809  15   34:2    69     END;
1810  15   34:1    71   CURSOR:=MAX(STUFFSTART,MAX(CURSOR-REPEATFACTOR,1));
1811  15   34:1    88   IF LINE<1 THEN CENTER;
1812  15   34:1    97   FINDXY(X,LINE);
1813  15   34:0    06 END (* LEFTMOVE *);
1814  15   34:0    20
1815  15   35:0     1 PROCEDURE RIGHTMOVE;
1816  15   35:0     1 VAR
1817  15   35:0     1   EOLPTR: PTRTYPE;
1818  15   35:0     0 BEGIN
1819  15   35:1     0   EOLPTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR;
1820  15   35:1    15   WHILE (EOLPTR<CURSOR+REPEATFACTOR) AND (EOLPTR<BUFCOUNT-1) DO
1821  15   35:2    28     BEGIN
1822  15   35:3    28       REPEATFACTOR:=REPEATFACTOR-(EOLPTR-CURSOR+1);
1823  15   35:3    37       CURSOR:=EOLPTR+1; (* BEGINNING OF THE LINE BELOW *)
1824  15   35:3    42       GETLEADING;
1825  15   35:3    45       CURSOR:=STUFFSTART;
1826  15   35:3    48       LINE:=LINE+1;
1827  15   35:3    56       IF LINE=SCREENHEIGHT+1 THEN SCROLLMARK:=LINESTART;
1828  15   35:3    69       EOLPTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR
1829  15   35:2    80     END;
1830  15   35:1    86   IF LINE>SCREENHEIGHT THEN
1831  15   35:2    93     IF (LINE-SCREENHEIGHT>=SCREENHEIGHT) OR (INDELETE) THEN
1832  15   35:3    06       CENTER
1833  15   35:2    06     ELSE
1834  15   35:3    10       SCROLLUP(SCROLLMARK,LINE-SCREENHEIGHT);
1835  15   35:1    20   CURSOR:=MIN(BUFCOUNT-1,CURSOR+REPEATFACTOR);
1836  15   35:1    33   FINDXY(X,LINE);
```

```
1837  15   35:0    42 END(* RIGHTMOVE *);
1838  15   35:0    56
1839  15   36:D     1 PROCEDURE LINEMOVE(REPEATFACTOR: INTEGER);
1840  15   36:D     2 VAR I: INTEGER;
1841  15   36:0     0 BEGIN
1842  15   36:1     0   I:=1;
1843  15   36:1     3   IF DIRECTION='<' THEN
1844  15   36:2     8     BEGIN
1845  15   36:3     8       WHILE (I<=REPEATFACTOR) AND (CURSOR>1) DO
1846  15   36:4    17         BEGIN
1847  15   36:5    17           IF EBUF^[CURSOR]=CHR(EOL) THEN CURSOR:=CURSOR-1; (* NULL LINE CASE *)
1848  15   36:5    30           CURSOR:=SCAN(-MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR; (* 1 UP *)
1849  15   36:5    46           IF CURSOR>=1 THEN BEGIN LINE:=LINE-1; I:=I+1 END;
1850  15   36:4    64         END;
1851  15   36:3    66       CURSOR:=MAX(1,CURSOR); (* BACK INTO REALITY *)
1852  15   36:3    75       ATEND:= (CURSOR=1);
1853  15   36:3    81       IF LINE<1 THEN CENTER
1854  15   36:2    88     END
1855  15   36:1    90   ELSE
1856  15   36:2    92     BEGIN (* DIRECTION='>' *)
1857  15   36:3    92       WHILE (I<=REPEATFACTOR) AND (CURSOR<BUFCOUNT-1) DO
1858  15   36:4    03         BEGIN
1859  15   36:5    03           CURSOR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR+1; (*1 DOWN *)
1860  15   36:5    20           IF CURSOR<BUFCOUNT THEN
1861  15   36:6    25             BEGIN
1862  15   36:7    25               I:=I+1; LINE:=LINE+1;
1863  15   36:7    38               IF LINE=SCREENHEIGHT+1 THEN SCROLLMARK:=CURSOR;
1864  15   36:6    51             END
1865  15   36:4    51         END;
1866  15   36:3    53       ATEND:= (CURSOR>=BUFCOUNT-1);
1867  15   36:3    61       IF LINE>SCREENHEIGHT THEN
1868  15   36:4    68         IF (LINE-SCREENHEIGHT>=SCREENHEIGHT) OR (COMMAND=PARAC)
1869  15   36:4    78            OR INREPLACE OR INDELETE
1870  15   36:4    83           THEN
1871  15   36:5    89             CENTER
1872  15   36:4    89           ELSE SCROLLUP(SCROLLMARK,LINE-SCREENHEIGHT);
1873  15   36:3    03       CURSOR:=MIN(CURSOR,BUFCOUNT-1)
1874  15   36:2    07     END;
1875  15   36:1    14   GETLEADING;
1876  15   36:1    17   CURSOR:=STUFFSTART; (* FORCED TO BEGINNING OF STUFF *)
1877  15   36:1    20   X:=BLANKS;
```

```
1878  15   36:0    24 END(* LINEMOVE *);
1879  15   36:0    40
1880  15   37:0     1 PROCEDURE JUMPBEGIN;
1881  15   37:0     0 BEGIN
1882  15   37:1     0   CURSOR:=1; CENTERCURSOR(TRASH,1,FALSE)
1883  15   37:0     8 END;
1884  15   37:0    24
1885  15   38:0     1 PROCEDURE JUMPEND;
1886  15   38:0     0 BEGIN
1887  15   38:1     0   CURSOR:=BUFCOUNT-1; CENTERCURSOR(TRASH,SCREENHEIGHT,FALSE)
1888  15   38:0    10 END;
1889  15   38:0    26
1890  15   39:0     1 PROCEDURE ADJUSTING;
1891  15   39:0     1 LABEL 1;
1892  15   39:0     1 TYPE
1893  15   39:0     1   MODES=(RELATIVE,LEFTJ,RIGHTJ,CENTER);
1894  15   39:0     1 VAR
1895  15   39:0     1   LLENGTH,TDELTA,I: INTEGER;
1896  15   39:0     4   SAVEDIR: CHAR;
1897  15   39:0     5   MODE: MODES;
1898  15   39:0     6
1899  15   40:0     1 PROCEDURE DOIT(DELTA:INTEGER);
1900  15   40:0     2 VAR
1901  15   40:0     2   EOLDIST: INTEGER;
1902  15   40:0     3   T: PACKED ARRAY [0..MAXSTRING] OF CHAR;
1903  15   40:0     0 BEGIN
1904  15   40:1     0   GETLEADING; (* SET LINESTART, STUFFSTART, AND BLANKS *)
1905  15   40:1     3   IF BLANKS+DELTA<0 THEN DELTA:=-BLANKS;
1906  15   40:1    14   IF (EBUF^[LINESTART]=CHR(DLE)) AND (STUFFSTART-LINESTART=2) THEN
1907  15   40:2    28     X:=ORD(EBUF^[LINESTART+1])+DELTA-32
1908  15   40:1    36   ELSE
1909  15   40:2    43     BEGIN
1910  15   40:3    43       IF STUFFSTART-LINESTART>2 THEN
1911  15   40:4    50         MOVELEFT(EBUF^[STUFFSTART],EBUF^[LINESTART+2],BUFCOUNT-STUFFSTART)
1912  15   40:3    63       ELSE
1913  15   40:4    65         BEGIN
1914  15   40:5    65           IF BUFCOUNT>BUFSIZE-100 THEN
1915  15   40:6    72             BEGIN
1916  15   40:7    72               ERROR('BUFFER OVERFLOW',NONFATAL);
1917  15   40:7    94               EXIT(ADJUSTING)
1918  15   40:6    98             END
```

```
1919  15   40:5   98          ELSE
1920  15   40:6   00            MOVERIGHT(EBUF^[STUFFSTART],EBUF^[LINESTART+2],BUFCOUNT-STUFFSTART);
1921  15   40:4   13        END;
1922  15   40:3   13      IF LINESTART+2<>STUFFSTART THEN
1923  15   40:4   20        BEGIN
1924  15   40:5   20          READJUST(LINESTART,LINESTART+2-STUFFSTART);
1925  15   40:5   29          BUFCOUNT:=BUFCOUNT+LINESTART+2-STUFFSTART;
1926  15   40:4   38        END;
1927  15   40:3   38      EBUF^[LINESTART]:=CHR(DLE);
1928  15   40:3   43      X:=BLANKS+DELTA;
1929  15   40:2   49    END;
1930  15   40:1   49  EBUF^[LINESTART+1]:=CHR(X+32);
1931  15   40:1   60  CURSOR:=LINESTART+2; GETLEADING;
1932  15   40:1   68  GOTOXY(0,LINE); ERASETOEOL(0,LINE); (* ERASE THE LINE *)
1933  15   40:1   82  LINEOUT(LINESTART,BYTES,BLANKS,LINE); GOTOXY(X,LINE);
1934  15   40:0   01 END(* DOIT *);
1935  15   40:0   14
1936  15   39:0    0 BEGIN (* ADJUSTING *)
1937  15   39:1    0   WITH PAGEZERO DO
1938  15   39:2    0     BEGIN
1939  15   39:3    0       SAVEDIR:=DIRECTION; EXITPROMPT:=FALSE; INDELETE:=FALSE; LASTPAT:=CURSOR;
1940  15   39:3   14       INREPLACE:=TRUE;
1941  15   39:3   18       PROMPTLINE:=ADJUSTPROMPT;
1942  15   39:3   26       PROMPT; NEEDPROMPT:=TRUE;
1943  15   39:3   33       MODE:=RELATIVE;
1944  15   39:3   36       SHOWCURSOR;
1945  15   39:3   39       FINDXY(X,LINE);
1946  15   39:3   48       TDELTA:=0;
1947  15   39:3   51       REPEAT
1948  15   39:4   51         CH:=GETCH;
1949  15   39:4   58         COMMAND:=MAPTOCOMMAND(CH);
1950  15   39:4   66         INFINITY:=FALSE;
1951  15   39:4   70         IF COMMAND=SLASHC THEN
1952  15   39:5   75           BEGIN
1953  15   39:6   75             REPEATFACTOR:=1; INFINITY:=TRUE; CH:=GETCH; COMMAND:=TRANSLATE[CH]
1954  15   39:5   95           END
1955  15   39:4   98         ELSE
1956  15   39:5   00           IF COMMAND=DIGIT THEN REPEATFACTOR:=GETNUM ELSE REPEATFACTOR:=1;
1957  15   39:4   17         IF COMMAND IN [UP,DOWN] THEN
1958  15   39:5   28           BEGIN
1959  15   39:6   28             IF COMMAND=UP THEN DIRECTION:='<' ELSE DIRECTION:='>';
```

```
1960  15   39:6    41          I:=1;
1961  15   39:6    44          ATEND:=FALSE;
1962  15   39:6    48          WHILE NOT ATEND AND ((I<=REPEATFACTOR) OR INFINITY) DO
1963  15   39:7    62            BEGIN
1964  15   39:8    62              I:=I+1;
1965  15   39:8    67              LINEMOVE(1);
1966  15   39:8    70              IF NOT ATEND THEN
1967  15   39:9    76                BEGIN
1968  15   39:0    76                  IF MODE=RELATIVE THEN DOIT(TDELTA)
1969  15   39:0    82                  ELSE
1970  15   39:1    86                    BEGIN
1971  15   39:2    86                      LLENGTH:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[STUFFSTART]);
1972  15   39:2    99                      CASE MODE OF
1973  15   39:2    02                        LEFTJ:  DOIT(LMARGIN-BLANKS);
1974  15   39:2    11                        RIGHTJ: DOIT((RMARGIN-LLENGTH+1)-BLANKS);
1975  15   39:2    24                        CENTER:
1976  15   39:3    24                        DOIT(((RMARGIN-LMARGIN+1)-LLENGTH) DIV 2-BLANKS+LMARGIN)
1977  15   39:2    43                      END (* CASE *)
1978  15   39:1    60                    END (* ELSE *)
1979  15   39:9    60                END; (* IF NOT ATEND *)
1980  15   39:7    60            END (* WHILE ... *)
1981  15   39:5    60        END
1982  15   39:4    62      ELSE
1983  15   39:5    64        IF COMMAND=LEFT THEN
1984  15   39:6    69          BEGIN
1985  15   39:7    69            DOIT(-REPEATFACTOR); TDELTA:=TDELTA-REPEATFACTOR; MODE:=RELATIVE
1986  15   39:6    78          END
1987  15   39:5    81        ELSE
1988  15   39:6    83          IF COMMAND=RIGHT THEN
1989  15   39:7    88            BEGIN
1990  15   39:8    88              DOIT(REPEATFACTOR); TDELTA:=TDELTA+REPEATFACTOR; MODE:=RELATIVE
1991  15   39:7    96            END
1992  15   39:6    99          ELSE
1993  15   39:7    01            IF COMMAND IN [LISTC,REPLACEC,COPYC] THEN
1994  15   39:8    09              BEGIN
1995  15   39:9    09                GETLEADING;
1996  15   39:9    12                LLENGTH:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[STUFFSTART]);
1997  15   39:9    25                IF COMMAND=LISTC THEN
1998  15   39:0    30                  BEGIN MODE:=LEFTJ; DOIT(LMARGIN-BLANKS) END
1999  15   39:9    40                ELSE
2000  15   39:0    42                  IF COMMAND=REPLACEC THEN
```

```
2001  15   39:1    47                      BEGIN MODE:=RIGHTJ; DOIT((RMARGIN-LLENGTH+1)-BLANKS) END
2002  15   39:0    61                    ELSE (* COMMAND=COPYC *)
2003  15   39:1    63                      BEGIN
2004  15   39:2    63                        MODE:=CENTER;
2005  15   39:2    66                        DOIT(((RMARGIN-LMARGIN+1)-LLENGTH) DIV 2-BLANKS+LMARGIN)
2006  15   39:1    85                      END
2007  15   39:8    87                  END
2008  15   39:7    87                ELSE
2009  15   39:8    89              IF CH<>CHR(ETX) THEN BEGIN ERRWAIT; SHOWCURSOR END;
2010  15   39:4    02        1: UNTIL CH=CHR(ETX);
2011  15   39:3    09        DIRECTION:=SAVEDIR;
2012  15   39:2    12      END;
2013  15   39:0    12 END;
2014  15   39:0    32
2015  15   41:D     3 FUNCTION TABBY: INTEGER;
2016  15   41:0     0 BEGIN
2017  15   41:1     0   IF REPEATFACTOR > 0 THEN
2018  15   41:2     5     IF DIRECTION = '>' THEN
2019  15   41:3    10       TABBY:=8*(REPEATFACTOR-1)+  8-X+ORD(ODD(X) AND ODD(248))
2020  15   41:2    28     ELSE
2021  15   41:3    33       BEGIN
2022  15   41:4    33         IF X=0 THEN TABBY:=REPEATFACTOR*8
2023  15   41:4    41         ELSE TABBY:=8*(REPEATFACTOR-1)+X-ORD(ODD(X-1) AND ODD(248))
2024  15   41:3    65       END
2025  15   41:1    68   ELSE TABBY:=0
2026  15   41:0    70 END;
2027  15   41:0    86
2028  15   42:D     1 PROCEDURE MOVING;
2029  15   42:D     1 VAR
2030  15   42:D     1   SAVEX: INTEGER;
2031  15   42:0     0 BEGIN
2032  15   42:1     0   INDELETE:=FALSE;
2033  15   42:1     4   INREPLACE:=FALSE;
2034  15   42:1     8   EXITPROMPT:=FALSE;
2035  15   42:1    12   IF INFINITY THEN
2036  15   42:2    17     BEGIN
2037  15   42:3    17       CASE COMMAND OF
2038  15   42:3    20         UP,LEFT: JUMPBEGIN;
2039  15   42:3    24         DOWN,RIGHT: JUMPEND;
2040  15   42:3    28         SPACE,ADVANCE,TAB: IF DIRECTION='<' THEN JUMPBEGIN ELSE JUMPEND
2041  15   42:3    37       END;
```

```
2042  15   42:3   66        NEEDPROMPT:=TRUE;
2043  15   42:3   70        NEXTCOMMAND;
2044  15   42:3   72        EXIT(MOVEIT)
2045  15   42:2   76      END;
2046  15   42:1   76    FINDXY(X,LINE);
2047  15   42:1   85    REPEAT
2048  15   42:2   85      OLDX:=X; OLDLINE:=LINE;
2049  15   42:2   97      CASE COMMAND OF
2050  15   42:2   00        LEFT: LEFTMOVE;
2051  15   42:2   04        RIGHT: RIGHTMOVE;
2052  15   42:2   08        SPACE: IF DIRECTION='<' THEN LEFTMOVE ELSE RIGHTMOVE;
2053  15   42:2   21        UP: UPMOVE;
2054  15   42:2   25        DOWN: DOWNMOVE;
2055  15   42:2   29        ADVANCE: LINEMOVE(REPEATFACTOR);
2056  15   42:2   34        PARAC:
2057  15   42:3   34          IF REPEATFACTOR>1000 THEN ERROR('TOO MANY',NONFATAL)
2058  15   42:3   53          ELSE LINEMOVE(SCREENHEIGHT*REPEATFACTOR);
2059  15   42:2   65        TAB: BEGIN
2060  15   42:4   65               IF REPEATFACTOR >= 4096 THEN
2061  15   42:5   72                 ERROR('INTEGER OVERFLOW',NONFATAL)
2062  15   42:4   92               ELSE
2063  15   42:5   97                 BEGIN
2064  15   42:6   97                   REPEATFACTOR:=TABBY;
2065  15   42:6   03                   IF DIRECTION='<' THEN LEFTMOVE ELSE RIGHTMOVE;
2066  15   42:6   14                   SAVEX:=X+1;
2067  15   42:6   21                   WHILE (X<>SAVEX) AND (X MOD 8<>0) DO
2068  15   42:7   36                     BEGIN
2069  15   42:8   36                       SAVEX:=X; REPEATFACTOR:=1;
2070  15   42:8   44                       IF DIRECTION='>' THEN RIGHTMOVE ELSE LEFTMOVE
2071  15   42:7   53                     END
2072  15   42:5   55                 END
2073  15   42:3   57             END
2074  15   42:2   57      END;
2075  15   42:2   02      IF EXITPROMPT OR (COMMAND=PARAC) THEN GOTOXY(X,LINE)
2076  15   42:2   20      ELSE
2077  15   42:3   22        IF LINE=OLDLINE THEN
2078  15   42:4   31        BEGIN
2079  15   42:5   31          IF X=OLDX+1 THEN
2080  15   42:6   42            GOTOXY(X,LINE) [KLUDGE FOR HAZELTINE TERMINALS THAT USED DLES]
2081  15   42:5   51          ELSE
2082  15   42:6   53            IF X=OLDX-1 THEN WRITE(CHR(BS))
```

```
2083  15   42:6    74            ELSE GOTOXY(X,LINE)
2084  15   42:4    85          END
2085  15   42:3    85          ELSE
2086  15   42:4    87            IF X=OLDX THEN
2087  15   42:5    96              BEGIN
2088  15   42:6    96                IF LINE=OLDLINE+1 THEN WRITE(CHR(LF))
2089  15   42:6    15                ELSE IF LINE=OLDLINE-1 THEN CONTROL(UPCURSOR)
2090  15   42:7    29                ELSE GOTOXY(X,LINE);
2091  15   42:5    43              END
2092  15   42:4    43            ELSE
2093  15   42:5    45              GOTOXY(X,LINE);
2094  15   42:2    54      REPEATFACTOR:=1;
2095  15   42:2    57      NEXTCOMMAND
2096  15   42:1    57    UNTIL NOT (COMMAND IN [UP,DOWN,LEFT,RIGHT,ADVANCE,SPACE,TAB]);
2097  15   42:1    71    IF EXITPROMPT THEN PROMPT;
2098  15   42:1    79    SHOWCURSOR;
2099  15   42:0    82  END (* MOVING *);
2100  15   42:0    04
2101  15   43:D     1  PROCEDURE PUTITBACK(C1,C2: PTRTYPE);
2102  15   43:D     3  VAR
2103  15   43:D     3    PTR: PTRTYPE;
2104  15   43:D     4    INDENT,LOFF: INTEGER;
2105  15   43:0     0  BEGIN
2106  15   43:1     0    PTR:=C1;
2107  15   43:1     3    WHILE PTR<=C2 DO
2108  15   43:2     8      BEGIN
2109  15   43:3     8        IF EBUF^[PTR]=CHR(EOL) THEN
2110  15   43:4    16          BEGIN
2111  15   43:5    16            PTR:=PTR+1; WRITELN;
2112  15   43:5    27            INDENT:=LEADBLANKS(PTR,LOFF);
2113  15   43:5    37            IF (PTR<C2) AND (INDENT>0) THEN
2114  15   43:6    46              WRITE(' ':INDENT);
2115  15   43:5    54            PTR:=PTR+LOFF
2116  15   43:4    55          END
2117  15   43:3    59        ELSE
2118  15   43:4    61          BEGIN WRITE(EBUF^[PTR]); PTR:=PTR+1 END;
2119  15   43:2    77      END;
2120  15   43:0    79  END;
2121  15   43:0    94
2122  15   30:D     1  PROCEDURE CLEAR(*X1,Y1,X2,Y2: INTEGER*);
2123  15   30:D     5  (* SCREEN CO-ORDINATE (X1,Y1) IS ASSUMED TO BE BEFORE (X2,Y2).  THIS
```

```
2124  15   30:D    5    PROCEDURE TAKES THESE CO-ORDINATES AND CLEARS (WRITES BLANKS) OVER
2125  15   30:D    5    THE SCREEN BETWEEN THEM (INCLUSIVE)  *)
2126  15   30:D    5 VAR XX,I: INTEGER;
2127  15   30:0    0 BEGIN
2128  15   30:1    0   GOTOXY(X1,Y1);
2129  15   30:1    5   XX:=X1;
2130  15   30:1    8   FOR I:=Y1 TO Y2-1 DO BEGIN IF I<>0 THEN ERASETOEOL(XX,I); XX:=0; WRITELN END
2131  15   30:1   47   IF Y1<>Y2 THEN FOR I:=0 TO X2 DO WRITE(' ')
2132  15   30:1   71   ELSE FOR I:=X1 TO X2 DO WRITE(' ')
2133  15   30:0   99 END;
2134  15   30:0   24
2135  15   44:D    1 PROCEDURE RESOLVESCREEN;
2136  15   44:D    1 VAR
2137  15   44:D    1   X1,X2,Y1,Y2,SAVE: INTEGER;
2138  15   44:D    6   C1,C2: PTRTYPE;
2139  15   44:0    0 BEGIN
2140  15   44:1    0   X1:=X;    Y1:=LINE;
2141  15   44:1   10   X2:=OLDX; Y2:=OLDLINE;
2142  15   44:1   20   IF NEWDIST>DIST THEN
2143  15   44:2   29     BEGIN C1:=CURSOR-1; C2:=OLDCURSOR; X1:=X1-1 END
2144  15   44:1   44   ELSE
2145  15   44:2   46     IF NEWDIST<DIST THEN
2146  15   44:3   55       BEGIN C2:=OLDCURSOR-1; C1:=CURSOR; X2:=X2-1 END
2147  15   44:2   70     ELSE
2148  15   44:3   72       EXIT(RESOLVESCREEN);
2149  15   44:1   76   IF (Y1>Y2) OR ((Y1=Y2) AND (X1>X2)) THEN
2150  15   44:2   89     BEGIN
2151  15   44:3   89       SAVE:=C1; C1:=C2; C2:=SAVE;
2152  15   44:3   98       SAVE:=Y1; Y1:=Y2; Y2:=SAVE;
2153  15   44:3   07       SAVE:=X1; X1:=X2; X2:=SAVE
2154  15   44:2   13     END;
2155  15   44:1   16   IF ABS(NEWDIST)>ABS(DIST) THEN
2156  15   44:2   27     CLEAR(X1,Y1,X2,Y2)
2157  15   44:1   31   ELSE
2158  15   44:2   35     BEGIN
2159  15   44:3   35       GOTOXY(X1,Y1);
2160  15   44:3   40       PUTITBACK(C1,C2)
2161  15   44:2   42     END;
2162  15   44:1   44   GOTOXY(X,LINE)
2163  15   44:0   53 END;
2164  15   44:0   66
```

```
2165  15   45:0     1 PROCEDURE DELETING;
2166  15   45:0     1 LABEL 1;
2167  15   45:0     1 VAR
2168  15   45:0     1   ATBOL,ANCHOR,SAVE: PTRTYPE;
2169  15   45:0     4   OK,ATBOT,NOMOVE: BOOLEAN;
2170  15   45:0     7   STARTLINE: INTEGER;
2171  15   45:0     8
2172  15   45:0     0 BEGIN
2173  15   45:1     0   DOFFSCREEN:=FALSE; INDELETE:=TRUE; INREPLACE:=FALSE; EXITPROMPT:=FALSE;
2174  15   45:1    16   ANCHOR:=CURSOR; NEWDIST:=0;
2175  15   45:1    23   GETLEADING; ATBOL:=LINESTART; ATBOT:=(CURSOR=STUFFSTART);
2176  15   45:1    34   PROMPTLINE:=DELETEPROMPT;
2177  15   45:1    42   PROMPT; NEEDPROMPT:=TRUE;
2178  15   45:1    49   SHOWCURSOR;
2179  15   45:1    52   FINDXY(X,LINE);
2180  15   45:1    61   STARTLINE:=LINE;
2181  15   45:1    66   REPEAT
2182  15   45:2    66     OLDCURSOR:=CURSOR;
2183  15   45:2    70     DIST:=NEWDIST;
2184  15   45:2    76     OLDX:=X; OLDLINE:=LINE;
2185  15   45:2    88     CH:=GETCH;
2186  15   45:2    95     COMMAND:=TRANSLATE[CH];
2187  15   45:2    04     IF COMMAND=DIGIT THEN REPEATFACTOR:=GETNUM ELSE REPEATFACTOR:=1;
2188  15   45:2    21     IF COMMAND IN [REVERSEC..DIGIT,ADVANCE,SPACE] THEN
2189  15   45:3    32       BEGIN
2190  15   45:4    32         CASE COMMAND OF
2191  15   45:4    35           LEFT: LEFTMOVE;
2192  15   45:4    39           RIGHT: RIGHTMOVE;
2193  15   45:4    43           SPACE: IF DIRECTION='<' THEN LEFTMOVE ELSE RIGHTMOVE;
2194  15   45:4    56           UP: UPMOVE;
2195  15   45:4    60           DOWN: DOWNMOVE;
2196  15   45:4    64           ADVANCE: LINEMOVE(REPEATFACTOR);
2197  15   45:4    69           REVERSEC,FORWARDC:
2198  15   45:5    69             BEGIN
2199  15   45:6    69               IF COMMAND=REVERSEC THEN
2200  15   45:7    74                 DIRECTION:='<'
2201  15   45:6    74               ELSE
2202  15   45:7    79                 DIRECTION:='>';
2203  15   45:6    82               GOTOXY(0,0); WRITE(DIRECTION); GOTOXY(X,LINE)
2204  15   45:5    04             END;
2205  15   45:4    06           TAB:
```

```
2206  15   45:5   06              BEGIN
2207  15   45:6   26                IF REPEATFACTOR>=4096 THEN ERROR('INTEGER OVFLW',NONFATAL)
2208  15   45:6   30                ELSE
2209  15   45:7   35                  BEGIN
2210  15   45:8   35                    REPEATFACTOR:=TABBY;
2211  15   45:8   41                    IF DIRECTION='<' THEN LEFTMOVE ELSE RIGHTMOVE
2212  15   45:7   50                  END
2213  15   45:5   52              END
2214  15   45:4   52          END;
2215  15   45:4   84          NEWDIST:=CURSOR-ANCHOR;
2216  15   45:4   90          RESOLVESCREEN;
2217  15   45:3   92        END
2218  15   45:2   92      ELSE
2219  15   45:3   94        IF (CH<>CHR(ESC)) AND (CH<>CHR(ETX)) THEN
2220  15   45:4   07          BEGIN ERRWAIT; GOTOXY(X,LINE) END
2221  15   45:1   19    UNTIL (CH IN [CHR(ETX),CHR(ESC)]);
2222  15   45:1   32    IF CH=CHR(ETX) THEN
2223  15   45:2   39      BEGIN
2224  15   45:3   39        GETLEADING; (* INDENTATION FIXUP *)
2225  15   45:3   42        IF ATBOT AND (CURSOR=STUFFSTART) THEN
2226  15   45:4   49          BEGIN CURSOR:=LINESTART; SAVE:=ANCHOR; ANCHOR:=ATBOL END;
2227  15   45:3   58        IF OKTODEL(CURSOR,ANCHOR) THEN
2228  15   45:4   67          BEGIN
2229  15   45:5   67            READJUST(MIN(CURSOR,ANCHOR),-ABS(CURSOR-ANCHOR));
2230  15   45:5   82            COPYLINE:=(CURSOR=LINESTART) AND ATBOT;
2231  15   45:5   90            IF ANCHOR<CURSOR THEN
2232  15   45:6   95              MOVELEFT(EBUF^[CURSOR],EBUF^[ANCHOR],BUFCOUNT-CURSOR)
2233  15   45:5   06            ELSE
2234  15   45:6   08              MOVELEFT(EBUF^[ANCHOR],EBUF^[CURSOR],BUFCOUNT-ANCHOR);
2235  15   45:5   19            BUFCOUNT:=BUFCOUNT-ABS(CURSOR-ANCHOR);
2236  15   45:5   27            CURSOR:=MIN(CURSOR,ANCHOR);
2237  15   45:5   36            GETLEADING; CURSOR:=MAX(STUFFSTART,CURSOR)
2238  15   45:4   41          END
2239  15   45:3   48        ELSE
2240  15   45:4   50          CURSOR:=SAVE
2241  15   45:2   50      END
2242  15   45:1   53    ELSE
2243  15   45:2   55      BEGIN
2244  15   45:3   55        COPYLINE:=FALSE; COPYOK:=TRUE;
2245  15   45:3   63        COPYSTART:=MIN(CURSOR,ANCHOR);
2246  15   45:3   73        COPYLENGTH:=ABS(CURSOR-ANCHOR);
```

```
2247  15    45:3     80        CURSOR:=ANCHOR;
2248  15    45:2     83      END;
2249  15    45:1     83    1:INDELETE:=FALSE;
2250  15    45:1     87    OK:=(LINE=STARTLINE) AND NOT DOFFSCREEN;
2251  15    45:1     99    UPSCREEN(OK,NOT OK,LINE);
2252  15    45:1     08    NEXTCOMMAND;
2253  15    45:0     10 END;
2254  15    45:0     28
2255  15    28:0      0 BEGIN
2256  15    28:1      0   IF COMMAND=DELETEC THEN
2257  15    28:2      5     DELETING
2258  15    28:1      5   ELSE
2259  15    28:2      9     IF COMMAND=ADJUSTC THEN
2260  15    28:3     14       BEGIN ADJUSTING; NEXTCOMMAND END
2261  15    28:2     18     ELSE MOVING;
2262  15    28:0     22 END;
2263  15    28:0     34
2264  15    28:0     34
2265  15    28:0     34
2266  15    28:0     34 (*$I MOVEIT    *)
2266  15    28:0     34 (*$I FIND      *)
2267  15     6:D      1 PROCEDURE FIND;
2268  15     6:D      1 VAR
2269  15     6:D      1   THERE,FOUND,LASTPATTERN: BOOLEAN;
2270  15     6:D      4   TRASH,COULDBE,PLENGTH,START,STOP,NEXTSTART: INTEGER;
2271  15     6:D     10   NEXT,PTR: PTRTYPE;
2272  15     6:D     12   MODE: (LITERAL,TOKEN);
2273  15     6:D     13   I: INTEGER;
2274  15     6:D     14   DELIMITER: CHAR;
2275  15     6:D     15   JUSTIN: BOOLEAN;
2276  15     6:D     16   POSSIBLE,PAT: PTYPE;
2277  15     6:D     44   USEOLD,VERIFY: BOOLEAN;
2278  15     6:D     46
2279  15    46:D      1 PROCEDURE NEXTCH;
2280  15    46:0      0 BEGIN
2281  15    46:1      0   CH:=GETCH;
2282  15    46:1      7   IF CH=CHR(ESC) THEN
2283  15    46:2     14       BEGIN
2284  15    46:3     14         IF NOT JUSTIN THEN REDISPLAY;
2285  15    46:3     23         SHOWCURSOR; NEXTCOMMAND;
2286  15    46:3     28         EXIT(FIND);
```

```
2287  15   46:2    32        END;
2288  15   46:1    32    IF (CH=CHR(EOL)) AND JUSTIN THEN
2289  15   46:2    41      BEGIN
2290  15   46:3    41        JUSTIN:=FALSE;
2291  15   46:3    45        BLANKCRT(1)
2292  15   46:2    46      END
2293  15   46:1    49    ELSE
2294  15   46:2    51      WRITE(CH);
2295  15   46:0    59 END;
2296  15   46:0    72
2297  15   47:D     1 PROCEDURE SKIP;
2298  15   47:0     0 BEGIN
2299  15   47:1     0   WHILE CH IN [CHR(SP),CHR(HT),CHR(EOL)] DO NEXTCH
2300  15   47:0    12 END;
2301  15   47:0    30
2302  15   48:D     1 PROCEDURE OPTIONS;
2303  15   48:0     0 BEGIN
2304  15   48:1     0   REPEAT
2305  15   48:2     0     CH:=UCLC(CH);
2306  15   48:2     8     IF CH='L' THEN
2307  15   48:3    13       BEGIN MODE:=LITERAL; NEXTCH END
2308  15   48:2    19     ELSE
2309  15   48:3    21       IF CH='V' THEN
2310  15   48:4    26         BEGIN VERIFY:=TRUE; NEXTCH END
2311  15   48:3    33       ELSE
2312  15   48:4    35         IF CH='T' THEN
2313  15   48:5    40           BEGIN MODE:=TOKEN; NEXTCH END;
2314  15   48:2    46     CH:=UCLC(CH);
2315  15   48:1    54   UNTIL NOT ((CH='V') OR (CH='T') OR (CH='L'));
2316  15   48:1    68   SKIP;
2317  15   48:1    70   IF (CH='S') OR (CH='S') THEN USEOLD:=TRUE;
2318  15   48:0    84 END;
2319  15   48:0    98
2320  15   49:D     1 PROCEDURE PARSESTRING(VAR PATTERN: PTYPE; VAR PLENGTH: INTEGER);
2321  15   49:D     3 VAR I,J: INTEGER;
2322  15   49:0     0 BEGIN
2323  15   49:1     0   SKIP;
2324  15   49:1     2   IF CH IN ['A'..'Z','A'..'Z','0'..'9',CHR(BS)] THEN
2325  15   49:2    31     BEGIN
2326  15   49:3    31       ERROR('INVALID DELIMITER.',NONFATAL);
2327  15   49:3    56       IF NOT JUSTIN THEN REDISPLAY;
```

```
2328  15   49:3    65       NEXTCOMMAND;
2329  15   49:3    67       EXIT(FIND);
2330  15   49:2    71     END;
2331  15   49:1    71   DELIMITER:=CH;
2332  15   49:1    75   I:=0;
2333  15   49:1    78   REPEAT
2334  15   49:2    78     NEXTCH;
2335  15   49:2    80     IF CH=CHR(BS) THEN
2336  15   49:3    87       BEGIN
2337  15   49:4    87         IF (PATTERN[I]<>CHR(EOL)) AND (I>0) THEN (* DON'T GO OVERBOARD! *)
2338  15   49:5    98           BEGIN
2339  15   49:6    98             WRITE(' ',CHR(BS));
2340  15   49:6    16             I:=I-1
2341  15   49:5    17           END
2342  15   49:4    21         ELSE CONTROL(RIGHTCURSOR); (* MAKE UP FOR THE <BS> NEXTCH WROTE OUT *)
2343  15   49:3    27       END
2344  15   49:2    27     ELSE
2345  15   49:3    29       BEGIN
2346  15   49:4    29         PATTERN[I]:=CH;
2347  15   49:4    33         I:=I+1
2348  15   49:3    34       END;
2349  15   49:1    38   UNTIL (CH=DELIMITER) OR (I>=MAXSTRING);
2350  15   49:1    49   IF I>=MAXCHAR THEN
2351  15   49:2    56     BEGIN
2352  15   49:3    56       ERROR('YOUR PATTERN IS TOO LONG',NONFATAL);
2353  15   49:3    87       IF NOT JUSTIN THEN REDISPLAY;
2354  15   49:3    96       NEXTCOMMAND; EXIT(FIND)
2355  15   49:2    02     END;
2356  15   49:1    02   PLENGTH:=I-1;
2357  15   49:0    07 END (* PARSESTRING *);
2358  15   49:0    22
2359  15   50:D     3 FUNCTION OK(PTR: PTRTYPE): BOOLEAN;
2360  15   50:D     4 (* COMPARE PAT AGAINST THE BUFFER *)
2361  15   50:D     4 VAR I: INTEGER;
2362  15   50:0     0 BEGIN
2363  15   50:1     0   I:=0;
2364  15   50:1     3   WHILE (I<PLENGTH) AND (EBUF^[PTR+I]=PAT[I]) DO I:=I+1;
2365  15   50:1    30   OK:= I=PLENGTH;
2366  15   50:0    37 END;
2367  15   50:0    52
2368  15   51:D     1 PROCEDURE SKIPKIND3(VAR CURSOR: PTRTYPE);
```

```
2369  15   51:0    0 BEGIN
2370  15   51:0    0   (* SKIP OVER KIND3 CHARACTERS IN THE EBUF.  UPDATE THE CURSOR
2371  15   51:0    0      TO THE FIRST NON-KIND3 CHARACTER                          *)
2372  15   51:1    0   WHILE EBUF^[CURSOR] IN [CHR(SP),CHR(HT),CHR(DLE),CHR(EOL)] DO
2373  15   51:2   19     IF EBUF^[CURSOR]=CHR(DLE) THEN CURSOR:=CURSOR+2
2374  15   51:2   31     ELSE CURSOR:=CURSOR+1;
2375  15   51:0   44 END;
2376  15   51:0   58
2377  15   52:D    1 PROCEDURE SCANBACKWARD;
2378  15   52:D    1 LABEL 1;
2379  15   52:D    1 VAR
2380  15   52:D    1   LOC: PTRTYPE;
2381  15   52:D    2   CHTHERE: BOOLEAN;
2382  15   52:0    0 BEGIN
2383  15   52:1    0   CHTHERE:=TRUE;
2384  15   52:1    3   THERE:=FALSE;
2385  15   52:1    7   FILLCHAR(PAT[0],SIZEOF(PAT),' ');
2386  15   52:1   17   MOVELEFT(TARGET[START],PAT[0],PLENGTH);
2387  15   52:1   32   WHILE CHTHERE AND NOT THERE DO
2388  15   52:2   40     BEGIN
2389  15   52:3   40       1: IF PTR>=PLENGTH THEN (* POSSIBLY THERE *)
2390  15   52:4   49         LOC:=SCAN(-PTR,=PAT[0],EBUF^[PTR])
2391  15   52:3   67       ELSE
2392  15   52:4   71         LOC:=-PTR;
2393  15   52:3   77       IF LOC=-PTR THEN (* NOT THERE! *)
2394  15   52:4   85         BEGIN
2395  15   52:5   85           CHTHERE:=FALSE; THERE:=FALSE
2396  15   52:4   88         END
2397  15   52:3   92       ELSE
2398  15   52:4   94         BEGIN
2399  15   52:5   94           PTR:=PTR+LOC; NEXT:=PTR-1;
2400  15   52:5   10           IF EBUF^[PTR-1]=CHR(DLE) THEN BEGIN PTR:=NEXT; GOTO 1 END;
2401  15   52:5   30           IF OK(PTR) THEN THERE:=TRUE ELSE PTR:=NEXT
2402  15   52:4   45         END
2403  15   52:2   51     END;
2404  15   52:0   53 END;
2405  15   52:0   70
2406  15   53:D    1 PROCEDURE SCANFORWARD;
2407  15   53:D    1 LABEL 1;
2408  15   53:D    1 VAR
2409  15   53:D    1   MAXSCAN,LOC: INTEGER;
```

```
2410  15   53:D    3   CHTHERE: BOOLEAN;
2411  15   53:0    0 BEGIN
2412  15   53:1    0   CHTHERE:=TRUE;
2413  15   53:1    3   THERE:=FALSE;
2414  15   53:1    7   FILLCHAR(PAT[0],SIZEOF(PAT),' ');
2415  15   53:1   17   MOVELEFT(TARGET[START],PAT[0],PLENGTH);
2416  15   53:1   32   WHILE CHTHERE AND NOT THERE DO
2417  15   53:2   40    BEGIN
2418  15   53:3   40     1: MAXSCAN:=(BUFCOUNT-PLENGTH)-PTR+1;
2419  15   53:3   53     IF MAXSCAN>0 THEN (* STILL STUFF TO SCAN *)
2420  15   53:4   58       LOC:=SCAN(MAXSCAN,=PAT[0],EBUF^[PTR])
2421  15   53:3   73     ELSE
2422  15   53:4   77       LOC:=MAXSCAN; (* DUMMY UP 'NOT FOUND' CONDITION *)
2423  15   53:3   80     IF LOC=MAXSCAN THEN
2424  15   53:4   85       BEGIN CHTHERE:=FALSE; THERE:=FALSE END
2425  15   53:3   92     ELSE
2426  15   53:4   94       BEGIN
2427  15   53:5   94         PTR:=LOC+PTR; NEXT:=PTR+1;
2428  15   53:5   10         IF EBUF^[PTR-1]=CHR(DLE) THEN BEGIN PTR:=NEXT; GOTO 1 END;
2429  15   53:5   30         IF OK(PTR) THEN THERE:=TRUE ELSE PTR:=NEXT
2430  15   53:4   45       END
2431  15   53:2   51    END;
2432  15   53:0   53 END;
2433  15   53:0   70
2434  15   54:D    1 PROCEDURE GOFORIT;
2435  15   54:D    1
2436  15   55:D    1 PROCEDURE NEXTLINE;
2437  15   55:D    1 (* GIVEN NEXTSTART, CALCULATE THE START AND STOP FOR THE NEXT LINE *)
2438  15   55:0    0 BEGIN
2439  15   55:1    0   LASTPATTERN:=FALSE;
2440  15   55:1    4   START:=NEXTSTART;
2441  15   55:1   10   STOP:=MIN(TLENGTH-1,START+SCAN(TLENGTH-START,=CHR(EOL),TARGET[START]));
2442  15   55:1   45   IF STOP=TLENGTH-1 THEN BEGIN STOP:=MAX(STOP,0); LASTPATTERN:=TRUE END;
2443  15   55:1   72   NEXTSTART:=STOP+1;
2444  15   55:0   80 END;
2445  15   55:0   92
2446  15   56:D    1 PROCEDURE NEXTTOKEN;
2447  15   56:D    1 (* GIVEN NEXTSTART, CALCULATE START AND STOP *)
2448  15   56:0    0 BEGIN
2449  15   56:1    0   LASTPATTERN:=FALSE;
2450  15   56:1    4   START:=NEXTSTART;
```

```
2451  15   56:1    10   (* SKIP OVER LEADING KIND3 CHARACTERS *)
2452  15   56:1    10   WHILE (TARGET[START] IN [CHR(SP),CHR(EOL),CHR(HT)]) AND (START<TLENGTH-1) DO
2453  15   56:2    38     START:=START+1;
2454  15   56:1    48   STOP:=START;
2455  15   56:1    54   (* GET THE NEXT TOKEN *)
2456  15   56:1    54   WHILE (KIND[TARGET[START]]=KIND[TARGET[STOP+1]]) AND (STOP<TLENGTH-1) DO
2457  15   56:2    93     STOP:=STOP+1;
2458  15   56:1    03   STOP:=MIN(STOP,TLENGTH-1);
2459  15   56:1    19   (* TO ACCURATELY TEST FOR THE LAST TOKEN, SCAN OFF THE TRAILING KIND3
2460  15   56:1    19      CHARACTERS *)
2461  15   56:1    19   NEXTSTART:=STOP+1;
2462  15   56:1    27   WHILE (TARGET[NEXTSTART] IN [CHR(EOL),CHR(SP),CHR(HT)]) AND
2463  15   56:1    43         (NEXTSTART<TLENGTH) DO NEXTSTART:=NEXTSTART+1;
2464  15   56:1    63   IF NEXTSTART=TLENGTH THEN BEGIN STOP:=MAX(STOP,0); LASTPATTERN:=TRUE END;
2465  15   56:0    88 END;
2466  15   56:0    06
2467  15   54:0     0 BEGIN(* GOFORIT *)
2468  15   54:1     0   FOUND:=FALSE;
2469  15   54:1     4   NEXT:=PTR;
2470  15   54:1    10   REPEAT
2471  15   54:2    10     PTR:=NEXT; (* SET TO NEXT PLACE TO SCAN FOR *)
2472  15   54:2    16     NEXTSTART:=0;  (* FOOL NEXTLINE INTO GIVING US START AND STOP FOR LINE 1 *)
2473  15   54:2    20     IF MODE=LITERAL THEN NEXTLINE ELSE NEXTTOKEN;
2474  15   54:2    33     PLENGTH:=STOP-START+1;
2475  15   54:2    45     IF DIRECTION='>' THEN SCANFORWARD ELSE SCANBACKWARD;
2476  15   54:2    56     IF THERE THEN
2477  15   54:3    61       BEGIN
2478  15   54:4    61         COULDBE:=PTR;
2479  15   54:4    67         FOUND:=TRUE;
2480  15   54:4    71         WHILE (NOT LASTPATTERN) AND FOUND DO
2481  15   54:5    81           BEGIN
2482  15   54:6    81             IF MODE=LITERAL THEN NEXTLINE ELSE NEXTTOKEN;
2483  15   54:6    94             PTR:=PTR+PLENGTH;
2484  15   54:6    04             SKIPKIND3(PTR); (* GO PAST THE JUNK ON THE NEXT LINE *)
2485  15   54:6    09             PLENGTH:=STOP-START+1; (* FOR THE NEW LINE *)
2486  15   54:6    21             FILLCHAR(PAT[0],SIZEOF(PAT),' ');
2487  15   54:6    31             MOVELEFT(TARGET[START],PAT[0],PLENGTH);
2488  15   54:6    46             IF PTR+PLENGTH > BUFCOUNT THEN
2489  15   54:7    57               FOUND:=FALSE
2490  15   54:6    57             ELSE
2491  15   54:7    63               IF NOT OK(PTR) THEN FOUND:=FALSE;
```

```
2492  15   54:5    77            END;
2493  15   54:3    79          END;
2494  15   54:3    79        (* IN TOKEN MODE MAKE SURE THE FIRST AND LAST CHARACTERS
2495  15   54:3    79           OF THE TARGET ARE ON 'TOKEN BOUNDARIES' *)
2496  15   54:2    79        IF MODE=TOKEN THEN IF KIND[PAT[0]]=ORD('A') THEN IF FOUND THEN
2497  15   54:5    05          BEGIN
2498  15   54:6    05            IF ((COULDBE>2) AND (EBUF^[COULDBE-2]<>CHR(DLE))) OR
2499  15   54:6    21               (COULDBE<=2) THEN (* WHEW! *)
2500  15   54:7    29              IF KIND[EBUF^[COULDBE]]=KIND[EBUF^[COULDBE-1]] THEN
2501  15   54:8    56                FOUND:=FALSE; (* FALSE FIND... DON'T COUNT IT. *)
2502  15   54:6    60            IF (PTR+PLENGTH<=BUFCOUNT-1) AND
2503  15   54:6    71               (KIND[EBUF^[PTR+PLENGTH-1]]=KIND[EBUF^[PTR+PLENGTH]]) THEN
2504  15   54:7    07              FOUND:=FALSE; (* ANOTHER FALSE FIND *)
2505  15   54:5    11          END;
2506  15   54:1    11      UNTIL FOUND OR NOT THERE;
2507  15   54:0    21    END(* GOFORIT *);
2508  15   54:0    38
2509  15   57:D     1    PROCEDURE PUTPROMPT(LEFT,RIGHT:STRING; REPEATFACTOR:INTEGER; LORT:BOOLEAN);
2510  15   57:0     0    BEGIN
2511  15   57:1     0      PROMPTLINE:=LEFT; PROMPT;
2512  15   57:1    20      WRITE('[');
2513  15   57:1    28      IF INFINITY THEN WRITE('/') ELSE WRITE(REPEATFACTOR);
2514  15   57:1    51      WRITE(']: ');
2515  15   57:1    64      IF LORT THEN IF MODE=TOKEN THEN WRITE('L(IT') ELSE WRITE('T(OK');
2516  15   57:1    04      WRITE(RIGHT)
2517  15   57:0    13    END;
2518  15   57:0    26
2519  15   58:D     1    PROCEDURE REPLACEIT;
2520  15   58:D     1    LABEL 1;
2521  15   58:0     0    BEGIN
2522  15   58:1     0      IF VERIFY THEN
2523  15   58:2     6        BEGIN
2524  15   58:3     6          CENTERCURSOR(TRASH,MIDDLE,NOT JUSTIN);
2525  15   58:3    19          PUTPROMPT(' REPLACE',',<ESC> ABORTS, ''R'' REPLACES, '' '' DOESN''T',
2526  15   58:3    72                    REPEATFACTOR-I+2,FALSE);
2527  15   58:3    82          SHOWCURSOR;
2528  15   58:3    85          CH:=GETCH;
2529  15   58:3    92          IF CH=CHR(ESC) THEN
2530  15   58:4    99            BEGIN
2531  15   58:5    99              GETLEADING; CURSOR:=MAX(CURSOR,STUFFSTART);
2532  15   58:5    11              NEXTCOMMAND; EXIT(FIND)
```

```
2533  15   58:4   17          END;
2534  15   58:3   17        IF (CH<>'R') AND (CH<>'R') THEN GOTO 1;
2535  15   58:2   28      END;
2536  15   58:2   28      (* REPLACE TARGET WITH SUBSTRING *)
2537  15   58:1   28        IF SLENGTH>CURSOR-LASTPAT THEN
2538  15   58:2   37          IF SLENGTH-(CURSOR-LASTPAT)+BUFCOUNT>BUFSIZE-200 THEN
2539  15   58:3   54              BEGIN
2540  15   58:4   54                ERROR('BUFFER FULL.  ABORTING REPLACE',NONFATAL);
2541  15   58:4   91                GETLEADING; CURSOR:=MAX(CURSOR,STUFFSTART);
2542  15   58:4   03                NEXTCOMMAND; EXIT(FIND);
2543  15   58:3   09              END
2544  15   58:2   09          ELSE
2545  15   58:3   11            MOVERIGHT(EBUF^[CURSOR],EBUF^[LASTPAT+SLENGTH],BUFCOUNT-CURSOR)
2546  15   58:1   26        ELSE
2547  15   58:2   28          IF SLENGTH<CURSOR-LASTPAT THEN
2548  15   58:3   37            MOVELEFT(EBUF^[CURSOR],EBUF^[LASTPAT+SLENGTH],BUFCOUNT-CURSOR);
2549  15   58:1   52        MOVELEFT(SUBSTRING[0],EBUF^[LASTPAT],SLENGTH);
2550  15   58:1   64        IF SLENGTH<>CURSOR-LASTPAT THEN
2551  15   58:2   73          READJUST(LASTPAT,SLENGTH-(CURSOR-LASTPAT));
2552  15   58:1   84        BUFCOUNT:=BUFCOUNT+SLENGTH-(CURSOR-LASTPAT);
2553  15   58:1   95        CURSOR  :=CURSOR  +SLENGTH-(CURSOR-LASTPAT);
2554  15   58:1   06        JUSTIN:=FALSE;
2555  15   58:1   10  1:END;
2556  15   58:1   24
2557  15    6:0    0  BEGIN
2558  15    6:1    0    JUSTIN:=TRUE;
2559  15    6:1    3    USEOLD:=FALSE;
2560  15    6:1    7    VERIFY:=FALSE;
2561  15    6:1   11    IF PAGEZERO.TOKDEF THEN MODE:=TOKEN ELSE MODE:=LITERAL;
2562  15    6:1   24    IF COMMAND=FINDC THEN
2563  15    6:2   29      PUTPROMPT(' FIND',' <TARGET>  =>',REPEATFACTOR,TRUE)
2564  15    6:1   55    ELSE
2565  15    6:2   59      PUTPROMPT(' REPLACE',' V(FY <TARG> <SUB>  =>',REPEATFACTOR,TRUE);
2566  15    6:1   99    NEEDPROMPT:=TRUE;
2567  15    6:1   03    NEXTCH; SKIP;
2568  15    6:1   07    OPTIONS;
2569  15    6:1   09    IF NOT USEOLD THEN
2570  15    6:2   15      BEGIN
2571  15    6:3   15        PARSESTRING(TARGET,TLENGTH);
2572  15    6:3   23        TDEFINED:=TRUE
2573  15    6:2   23      END;
```

```
2574  15    6:1    27   IF COMMAND=REPLACEC THEN
2575  15    6:2    32     BEGIN
2576  15    6:3    32       NEXTCH; SKIP;
2577  15    6:3    36       USEOLD:=FALSE;
2578  15    6:3    40       OPTIONS;
2579  15    6:3    42       IF NOT USEOLD THEN
2580  15    6:4    48         BEGIN
2581  15    6:5    48           PARSESTRING(SUBSTRING,SLENGTH);
2582  15    6:5    56           SDEFINED:=TRUE
2583  15    6:4    56         END
2584  15    6:2    60     END;
2585  15    6:1    60   HOME;
2586  15    6:1    63   CLEARLINE(0);
2587  15    6:1    67   IF ((COMMAND=FINDC) AND TDEFINED)
2588  15    6:1    74      OR ((COMMAND=REPLACEC) AND SDEFINED AND TDEFINED) THEN
2589  15    6:2    88     BEGIN
2590  15    6:3    88       I:=1;
2591  15    6:3    91       FOUND:=TRUE;
2592  15    6:3    94       PTR:=CURSOR;
2593  15    6:3    97       WHILE ((I<=REPEATFACTOR) OR INFINITY) AND FOUND DO
2594  15    6:4    08         BEGIN
2595  15    6:5    08           GOFORIT; (* FIND THE TARGET (HANDLES TOKEN AND LITERAL MODE) *)
2596  15    6:5    10           I:=I+1;
2597  15    6:5    15           IF FOUND THEN
2598  15    6:6    18             BEGIN
2599  15    6:7    18               CURSOR:=PTR+PLENGTH; LASTPAT:=COULDBE; (*SET UP FOR NEXT TIME*)
2600  15    6:7    26               IF COMMAND=REPLACEC THEN REPLACEIT;
2601  15    6:7    33               IF DIRECTION='<' THEN PTR:=COULDBE-1 ELSE PTR:=CURSOR;
2602  15    6:6    48             END;
2603  15    6:4    48         END;
2604  15    6:3    50       IF NOT FOUND THEN
2605  15    6:4    54         IF NOT( INFINITY AND (I>2) ) THEN
2606  15    6:5    64           ERROR('PATTERN NOT IN THE FILE',NONFATAL)
2607  15    6:2    91     END
2608  15    6:1    94   ELSE
2609  15    6:2    96     ERROR('NO OLD PATTERN.',NONFATAL);
2610  15    6:1    18   CENTERCURSOR(TRASH,MIDDLE,NOT JUSTIN);
2611  15    6:1    28   GETLEADING;
2612  15    6:1    31   CURSOR:=MAX(STUFFSTART,CURSOR);
2613  15    6:1    40   SHOWCURSOR;
2614  15    6:1    43   NEXTCOMMAND
```

```
2615  15    6:0    43 END;
2616  15    6:0    60 (*$I FIND      *)
2616  15    6:0    60 (*$I USER      *)
2617  15    2:D     1 PROCEDURE NEXTCOMMAND;
2618  15    2:0     0 BEGIN
2619  15    2:1     0   IF NEEDPROMPT THEN
2620  15    2:2     5     BEGIN
2621  15    2:3     5       PROMPTLINE:=COMPROMPT; {MADE VARIABLE FOR SCREENS OF SHORT WIDTH. MAB}
2622  15    2:3    13       PROMPT;
2623  15    2:3    16       NEEDPROMPT:=FALSE;
2624  15    2:3    20       SHOWCURSOR
2625  15    2:2    20     END;
2626  15    2:1    23   CH:=GETCH;
2627  15    2:1    30   COMMAND:=MAPTOCOMMAND(CH);
2628  15    2:0    38 END(* NEXTCOMMAND *);
2629  15    2:0    50
2630  15   59:D     1 PROCEDURE COMMANDER;
2631  15   59:0     0 BEGIN
2632  15   59:1     0   INFINITY:=FALSE;
2633  15   59:1     4   IF COMMAND=SLASHC THEN
2634  15   59:2     9     BEGIN REPEATFACTOR:=1; INFINITY:=TRUE; NEXTCOMMAND END
2635  15   59:1    18   ELSE
2636  15   59:2    20     IF COMMAND=DIGIT THEN REPEATFACTOR:=GETNUM ELSE REPEATFACTOR:=1;
2637  15   59:1    37   CASE COMMAND OF
2638  15   59:1    40     ILLEGAL: BEGIN ERRWAIT; SHOWCURSOR; NEXTCOMMAND END;
2639  15   59:1    50     REVERSEC,FORWARDC: FIXDIRECTION;
2640  15   59:1    54     COPYC: COPY;
2641  15   59:1    58     DUMPC: DUMP;
2642  15   59:1    62     FINDC: FIND;
2643  15   59:1    66     INSERTC: INSERTIT;
2644  15   59:1    70     JUMPC: JUMP;
2645  15   59:1    74     LISTC: NEXTCOMMAND; (* NOT YET, DEPENDS ON TERAK PAN *)
2646  15   59:1    78     MACRODEFC: DEFMACRO;
2647  15   59:1    82     QUITC: ; (* EXIT HANDLED IN OUTER BLOCK *)
2648  15   59:1    84     REPLACEC: FIND;
2649  15   59:1    88     SETC: SETSTUFF;
2650  15   59:1    92     VERIFYC: VERIFY;
2651  15   59:1    96     XECUTEC: XMACRO;
2652  15   59:1    00     ZAPC: ZAPIT;
2653  15   59:1    04     EQUALC: BEGIN
2654  15   59:3    04       CURSOR:=LASTPAT;
```

```
2655  15   59:3    07        GETLEADING;
2656  15   59:3    10        CURSOR:=MAX(CURSOR,STUFFSTART);
2657  15   59:3    19        CENTERCURSOR(TRASH,MIDDLE,FALSE);
2658  15   59:3    29        SHOWCURSOR; NEXTCOMMAND
2659  15   59:2    32      END;
2660  15   59:1    36      ADJUSTC,DELETEC,PARAC,UP,DOWN,LEFT,RIGHT,ADVANCE,TAB,SPACE: MOVEIT
2661  15   59:1    36    END (* BIG LONG CASE STATEMENT *);
2662  15   59:0    04  END (* COMMANDER *);
2663  15   59:0    18
2664  15    1:0     0  BEGIN (* EDITCORE *)
2665  15    1:1     0    NEXTCOMMAND;
2666  15    1:1     2    WHILE COMMAND<>QUITC DO COMMANDER
2667  15    1:0     7  END;
2668  15    1:0    26
2669  15    1:0    26
2670  15    1:0    26  (*$I USER       *)
2670  15    1:0    26  (*$I MISC       *)
2671   1   12:D     3  FUNCTION MIN(* (A,B:INTEGER):INTEGER *);
2672   1   12:0     0  BEGIN
2673   1   12:1     0    IF A<B THEN MIN:=A ELSE MIN:=B
2674   1   12:0    10  END;
2675   1   12:0    26
2676   1   13:D     3  FUNCTION MAX (*(A,B:INTEGER):INTEGER*);
2677   1   13:0     0  BEGIN
2678   1   13:1     0    IF A>B THEN MAX:=A ELSE MAX:=B
2679   1   13:0    10  END;
2680   1   13:0    26
2681   1    4:D     3  FUNCTION GETCH(*:CHAR*);
2682   1    4:D     3  VAR GCH: CHAR;
2683   1    4:0     0  BEGIN
2684   1    4:1     0    READ(KEYBOARD,GCH);
2685   1    4:1     8    IF EOLN(KEYBOARD) THEN GCH:=CHR(EOL);
2686   1    4:1    21    GETCH:=GCH;
2687   1    4:0    24  END;
2688   1    4:0    36
2689   1   16:D     1  PROCEDURE CONTROL(*(WHAT: SCREENCOMMAND)*);
2690   1   16:0     0  BEGIN
2691   1   16:1     0    WITH SCREEN DO
2692   1   16:2     0      BEGIN
2693   1   16:3     0        IF HASPREFIX[WHAT] THEN WRITE(PREFIX);
2694   1   16:3    20        WRITE(CH[WHAT]);
```

```
2695   1   16:3    32        WRITE(FILLIT); [SO THAT THE SLOWER TERMINALS CAN KEEP UP<--M. BERNARD]
2696   1   16:2    41      END
2697   1   16:0    41  END;
2698   1   16:0    54
2699   1   14:0     3  FUNCTION SCREENHAS(*(WHAT: SCREENCOMMAND): BOOLEAN*);
2700   1   14:0     0  BEGIN
2701   1   14:1     0    SCREENHAS:=SCREEN.CH[WHAT]<>CHR(0);
2702   1   14:0     9  END;
2703   1   14:0    22
2704   1   15:0     3  FUNCTION HASKEY(*(WHAT: KEYCOMMAND): BOOLEAN*);
2705   1   15:0     0  BEGIN
2706   1   15:1     0    HASKEY:=KEYBRD.CH[WHAT] <> CHR(0);
2707   1   15:0     9  END;
2708   1   15:0    22
2709   1   33:0     3  FUNCTION MAPCRTCOMMAND(VAR KCH:CHAR): KEYCOMMAND;
2710   1   33:0     4  VAR WHATITIS: KEYCOMMAND;
2711   1   33:0     5      PREFIXREAD: BOOLEAN;
2712   1   33:0     0  BEGIN
2713   1   33:1     0    WITH KEYBRD DO
2714   1   33:2     0    BEGIN
2715   1   33:3     0      IF (KCH=PREFIX) AND (PREFIX <> CHR(0)) THEN
2716   1   33:4    14        BEGIN
2717   1   33:5    14          PREFIXREAD:=TRUE;
2718   1   33:5    17          READ(KEYBOARD,KCH);
2719   1   33:4    24        END
2720   1   33:3    24      ELSE
2721   1   33:4    26        PREFIXREAD:=FALSE;
2722   1   33:3    29      WHATITIS:=BACKSPACEKEY;
2723   1   33:3    32      WHILE (WHATITIS <> NOTLEGAL) AND NOT((CH[WHATITIS]=KCH) AND
2724   1   33:3    43            (PREFIXREAD=HASPREFIX[WHATITIS])) DO
2725   1   33:4    59        WHATITIS:=SUCC(WHATITIS);
2726   1   33:3    66      MAPCRTCOMMAND:=WHATITIS;
2727   1   33:2    69    END;
2728   1   33:0    69  END;
2729   1   33:0    84
2730   1    8:0     3  FUNCTION MAPTOCOMMAND(* (CH:CHAR): COMMANDS *);
2731   1    8:0     4  (* FOR NOW, ONLY THE VECTOR KEYS GO THROUGH THE NEW KEYBOARD RECORD *)
2732   1    8:0     4  VAR KCMD: KEYCOMMAND;
2733   1    8:0     0  BEGIN
2734   1    8:1     0    IF (CH=KEYBRD.PREFIX) AND (CH<>CHR(0)) THEN
2735   1    8:2    11      BEGIN
```

```
2736   1     8:3     11        KCMD:=MAPCRTCOMMAND(CH);
2737   1     8:3     19        IF KCMD IN [UPKEY..RIGHTKEY] THEN
2738   1     8:4     27          CASE KCMD OF
2739   1     8:4     30            UPKEY:     MAPTOCOMMAND:=UP;
2740   1     8:4     35            DOWNKEY:   MAPTOCOMMAND:=DOWN;
2741   1     8:4     40            LEFTKEY:   MAPTOCOMMAND:=LEFT;
2742   1     8:4     45            RIGHTKEY:  MAPTOCOMMAND:=RIGHT
2743   1     8:4     45          END
2744   1     8:2     66      END
2745   1     8:1     66    ELSE
2746   1     8:2     68      MAPTOCOMMAND:=TRANSLATE[CH]
2747   1     8:0     74  END;
2748   1     8:0     90
2749   1     9:D      3  FUNCTION UCLC(*(CH:CHAR):CHAR*); (* MAP LOWER CASE TO UPPER CASE *)
2750   1     9:0      0  BEGIN
2751   1     9:1      0    IF CH IN ['A'..'Z'] THEN UCLC:=CHR(ORD(CH)-32) ELSE UCLC:=CH
2752   1     9:0     31  END;
2753   1     9:0     46
2754   1    10:D      1  PROCEDURE PROMPT;
2755   1    10:0      0  BEGIN
2756   1    10:1      0    PROMPTLINE[1]:=DIRECTION;
2757   1    10:1      6    SAVETOP:=PROMPTLINE;
2758   1    10:1     14    CONTROL(WHOME);
2759   1    10:1     17    CLEARLINE(0);
2760   1    10:1     20    WRITE(PROMPTLINE)
2761   1    10:0     30  END;
2762   1    10:0     42
2763   1     5:D      1  PROCEDURE CLEARSCREEN;
2764   1     5:D      1  VAR I:INTEGER;
2765   1     5:0      0  BEGIN
2766   1     5:1      0    IF SCREENHAS(CLEARSCN) THEN
2767   1     5:2      7      CONTROL(CLEARSCN)
2768   1     5:1      8    ELSE
2769   1     5:2     12      BEGIN
2770   1     5:3     12        HOME;
2771   1     5:3     14        ERASEOS(0,0)
2772   1     5:2     16      END;
2773   1     5:0     18  END;
2774   1     5:0     30
2775   1     7:D      1  PROCEDURE CLEARLINE(*Y:INTEGER*);
2776   1     7:D      2  VAR I: INTEGER;
```

```
2777   1     7:0      0 BEGIN
2778   1     7:1      0   IF SCREENHAS(CLEARLNE) THEN
2779   1     7:2      7     CONTROL(CLEARLNE)
2780   1     7:1      8   ELSE
2781   1     7:2     12     BEGIN
2782   1     7:3     12       GOTOXY(0,Y);
2783   1     7:3     17       ERASETOEOL(0,Y);
2784   1     7:2     21     END;
2785   1     7:0     21 END;
2786   1     7:0     34
2787   1    17:D      1 PROCEDURE PUTMSG;
2788   1    17:0      0 BEGIN
2789   1    17:1      0   CONTROL(WHOME);
2790   1    17:1      3   CLEARLINE(0);
2791   1    17:1      6   SAVETOP:=MSG;
2792   1    17:1     14   WRITE(MSG);
2793   1    17:0     24 END;
2794   1    17:0     36
2795   1    18:D      1 PROCEDURE HOME;
2796   1    18:0      0 BEGIN
2797   1    18:1      0   IF SCREENHAS(WHOME) THEN
2798   1    18:2      7     CONTROL(WHOME)
2799   1    18:1      8   ELSE
2800   1    18:2     12     GOTOXY(0,0);
2801   1    18:0     17 END;
2802   1    18:0     30
2803   1     3:D      1 PROCEDURE ERASETOEOL(*X,LINE:INTEGER*);
2804   1     3:D      3 VAR I: INTEGER;
2805   1     3:0      0 BEGIN
2806   1     3:1      0   IF SCREENHAS(ERASEEOL) THEN CONTROL(ERASEEOL)
2807   1     3:1      8   ELSE
2808   1     3:2     12     BEGIN
2809   1     3:3     12       IF LINE=SCREENHEIGHT THEN UNITWRITE(2,BLANKAREA,SCREENWIDTH-X)
2810   1     3:3     29       ELSE UNITWRITE(2,BLANKAREA,SCREENWIDTH-X+1);
2811   1     3:3     45       GOTOXY(X,LINE)
2812   1     3:2     50     END;
2813   1     3:0     50 END;
2814   1     3:0     62
2815   1    20:D      1 PROCEDURE BLANKCRT(*Y: INTEGER*);
2816   1    20:0      0 BEGIN
2817   1    20:1      0   IF SCREENHAS(ERASEEOS) THEN BEGIN GOTOXY(0,Y); CONTROL(ERASEEOS) END
```

```
2818   1   20:1    15   ELSE
2819   1   20:2    17     IF Y=1 THEN
2820   1   20:3    22       BEGIN
2821   1   20:4    22         CLEARSCREEN;
2822   1   20:4    24         WRITELN(SAVETOP)
2823   1   20:3    40       END
2824   1   20:2    40     ELSE
2825   1   20:3    42       BEGIN
2826   1   20:4    42         GOTOXY(0,Y);
2827   1   20:4    47         ERASEOS(0,Y);
2828   1   20:3    51       END;
2829   1   20:0    51 END;
2830   1   20:0    64
2831   1    6:D     1 PROCEDURE ERASEOS(*X,LINE*);
2832   1    6:D     3 VAR I: INTEGER;
2833   1    6:0     0 BEGIN
2834   1    6:1     0   IF SCREENHAS(ERASEEOS) THEN
2835   1    6:2     7     CONTROL(ERASEEOS)
2836   1    6:1     8   ELSE
2837   1    6:2    12     BEGIN
2838   1    6:3    12       ERASETOEOL(X,LINE);
2839   1    6:3    16       FOR I:=LINE+1 TO SCREENHEIGHT DO BEGIN WRITELN; CLEARLINE(I) END;
2840   1    6:3    45       GOTOXY(X,LINE);
2841   1    6:2    50     END;
2842   1    6:0    50 END;
2843   1    6:0    64
2844   1   19:D     1 PROCEDURE ERRWAIT;
2845   1   19:0     0 BEGIN
2846   1   19:1     0   WRITE(CHR(BELL));
2847   1   19:1     8   PROMPT;
2848   1   19:0    10 END;
2849   1   19:0    22
2850   1   19:0    22
2851   1    2:D     1 PROCEDURE ERROR(*S: STRING;HOWBAD: ERRORTYPE*);
2852   1    2:0     0 BEGIN
2853   1    2:1     0   UNITCLEAR(1); (* THROW AWAY ALL CHARACTERS QUEUED UP *)
2854   1    2:1     8   IF HOWBAD=FATAL THEN
2855   1    2:2    13     BLANKCRT(1)
2856   1    2:1    14   ELSE
2857   1    2:2    18     BEGIN HOME; CLEARLINE(0) END;
2858   1    2:1    23   WRITE('ERROR: ',S);
```

```
2859  1   2:1   49   IF HOWBAD=FATAL THEN
2860  1   2:2   54     EXIT(EDITOR)
2861  1   2:1   58   ELSE
2862  1   2:2   60     BEGIN
2863  1   2:3   60       WRITE('  PLEASE PRESS <SPACEBAR> TO CONTINUE.');
2864  1   2:3   08       REPEAT UNTIL GETCH=' '; NEEDPROMPT:=TRUE
2865  1   2:2   16     END;
2866  1   2:0   20 END;
2867  1   2:0   34 (*$I MISC       *)
2867  1   2:0   34 (*$I UTIL       *)
2868  1  21:D    3 FUNCTION LEADBLANKS(* (PTR: PTRTYPE; VAR BYTES: INTEGER): INTEGER *);
2869  1  21:D    5 (* ON ENTRY-
2870  1  21:D    5       PTR POINTS TO THE BEGINNING OF A LINE
2871  1  21:D    5    ON EXIT-
2872  1  21:D    5       FUNCTION RETURNS THE NUMBER OF LEADING BLANKS ON THAT LINE.
2873  1  21:D    5       BYTES HAS THE OFFSET INTO THE LINE OF THE FIRST NON-BLANK CHARACTER *)
2874  1  21:D    5 VAR
2875  1  21:D    5   OLDPTR: PTRTYPE;
2876  1  21:D    6   INDENT: INTEGER;
2877  1  21:0    0 BEGIN
2878  1  21:1    0   OLDPTR:=PTR; INDENT:=0;
2879  1  21:1    6   WHILE ORD(EBUF^[PTR]) IN [CHT,SP,DLE] DO
2880  1  21:2   22     BEGIN
2881  1  21:3   22       IF EBUF^[PTR]=CHR(DLE) THEN
2882  1  21:4   30         BEGIN PTR:=PTR+1; INDENT:=INDENT+ORD(EBUF^[PTR])-32 END
2883  1  21:3   45       ELSE
2884  1  21:4   47         IF ORD(EBUF^[PTR])=SP THEN INDENT:=INDENT+1
2885  1  21:4   56         ELSE
2886  1  21:5   62           (*HT*) INDENT:=((INDENT DIV 8)+1)*8;  (* KLUDGE FOR COLUMNAR TAB! *)
2887  1  21:3   71       PTR:=PTR+1
2888  1  21:2   72     END;
2889  1  21:1   78   BYTES:=PTR-OLDPTR;
2890  1  21:1   83   LEADBLANKS:=INDENT;
2891  1  21:0   86 END(*LEADBLANKS*);
2892  1  21:0   00
2893  1  11:D    1 PROCEDURE REDISPLAY;
2894  1  11:D    1 (* DO A TOTAL UPDATE OF THE SCREEN.  NOTE THAT THIS CODE IS PARTIALLY A
2895  1  11:D    1    DUPLICATE OF LINEOUT/UPSCREEN FOR REASONS OF SPEED.  THIS PROCEDURE IS
2896  1  11:D    1    CALLED ONLY FROM CENTERCURSOR *)
2897  1  11:D    1 VAR
2898  1  11:D    1   LINEDIST,EOLDIST,LINE: INTEGER;
```

```
2899   1   11:0     4   PTR: PTRTYPE;
2900   1   11:0     5   T: PACKED ARRAY [0..MAXSW] OF CHAR;
2901   1   11:0     0 BEGIN
2902   1   11:1     0   BLANKCRT(1);
2903   1   11:1     3   LINE:=1;
2904   1   11:1     6   PTR:=LINE1PTR;
2905   1   11:1    11   REPEAT
2906   1   11:2    11     BLANKS:=MIN(LEADBLANKS(PTR,BYTES),SCREENWIDTH);
2907   1   11:2    25     GOTOXY(BLANKS,LINE);
2908   1   11:2    30     PTR:=PTR+BYTES;
2909   1   11:2    35     EOLDIST:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR]);
2910   1   11:2    48     LINEDIST:=MAX(0,MIN(EOLDIST,SCREENWIDTH-BLANKS+1));
2911   1   11:2    65     MOVELEFT(EBUF^[PTR],T[0],LINEDIST);
2912   1   11:2    74     IF EBUF^[PTR+LINEDIST]<>CHR(EOL) THEN (* LINE TRUNCATION *)
2913   1   11:3    84       T[MAX(0,LINEDIST-1)]:='!';
2914   1   11:2    96     WRITE(T:LINEDIST);
2915   1   11:2    06     PTR:=PTR+EOLDIST+1; LINE:=LINE+1
2916   1   11:1    14   UNTIL (LINE>SCREENHEIGHT) OR (PTR>=BUFCOUNT)
2917   1   11:0    24 END;
2918   1   11:0    42
2919   1    1:0     1 PROCEDURE CENTERCURSOR
2920   1   22:0     4 (*VAR LINE: INTEGER; LINESUP: INTEGER; NEWSCREEN: BOOLEAN*);
2921   1   22:0     4 (* FIGURE OUT IF THE CURSOR IS STILL ON THE SCREEN.  IF IT IS, AND
2922   1   22:0     4    NEWSCREEN IS FALSE, THEN NO REDISPLAY IS DONE.  OTHERWISE AN ATTEMPT
2923   1   22:0     4    IS MADE TO POSITION THE CURSOR AT LINE "LINESUP".  LINE IS THEN UPDATED
2924   1   22:0     4    TO THE ACTUAL LINE THE CURSOR WAS FORCED TO. *)
2925   1   22:0     4 VAR
2926   1   22:0     4   MARK: INTEGER;
2927   1   22:0     5   PTR: PTRTYPE;
2928   1   22:0     0 BEGIN
2929   1   22:1     0   IF EBUF^[CURSOR]=CHR(EOL) THEN PTR:=CURSOR ELSE PTR:=CURSOR+1;
2930   1   22:1    18   LINE:=0;
2931   1   22:1    21   REPEAT
2932   1   22:2    21     PTR:=PTR-1;
2933   1   22:2    26     PTR:=SCAN(-MAXCHAR,=CHR(EOL),EBUF^[PTR])+PTR;
2934   1   22:2    42     LINE:=LINE+1;
2935   1   22:2    48     IF LINE=LINESUP THEN MARK:=PTR;
2936   1   22:1    57   UNTIL (LINE>SCREENHEIGHT) OR ((LINE1PTR=PTR+1) AND NOT NEWSCREEN) OR (PTR<1)
2937   1   22:1    78   IF LINE>SCREENHEIGHT THEN (* OFF THE SCREEN *)
2938   1   22:2    84     BEGIN LINE1PTR:=MARK+1; REDISPLAY; LINE:=LINESUP END
2939   1   22:1    95   ELSE
```

```
2940   1   22:2    97      IF LINE1PTR=PTR+1 THEN
2941   1   22:3    06        BEGIN
2942   1   22:4    06          IF NEWSCREEN THEN REDISPLAY
2943   1   22:3    09        END
2944   1   22:2    11      ELSE
2945   1   22:3    13        BEGIN
2946   1   22:4    13          LINE1PTR:=1; REDISPLAY
2947   1   22:3    17        END;
2948   1   22:0    19 END;
2949   1   22:0    34
2950   1   23:D     1 PROCEDURE FINDXY(*VAR INDENT,LINE: INTEGER*);
2951   1   23:D     3 VAR
2952   1   23:D     3   I,LEAD: INTEGER;
2953   1   23:D     5   PTR,EOLPTR: PTRTYPE;
2954   1   23:0     0 BEGIN
2955   1   23:0     0   (* PLACE CRT CURSOR ON THE SCREEN AT THE POSITION CORRESPONDING
2956   1   23:0     0      TO THE LOGICAL CURSOR. *)
2957   1   23:1     0   LINE:=1;
2958   1   23:1     3   PTR:=LINE1PTR;
2959   1   23:1     8   EOLPTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR])+PTR;
2960   1   23:1    23   WHILE EOLPTR<CURSOR DO
2961   1   23:2    28     BEGIN
2962   1   23:3    28       LINE:=LINE+1; PTR:=EOLPTR+1; (* SET UP FOR THE NEXT LINE *)
2963   1   23:3    39       EOLPTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR])+PTR
2964   1   23:2    50     END;
2965   1   23:2    56   (* NOW FIND THE INDENTATION ON THAT LINE OF THE CURSOR *)
2966   1   23:1    56   LEAD:=LEADBLANKS(PTR,I);
2967   1   23:1    65   INDENT:=MIN(SCREENWIDTH,(LEAD-I)+(CURSOR-PTR));
2968   1   23:1    79                              (* (EXTRA SPACES) + (OFFSET INTO LINE) *)
2969   1   23:0    79 END;(* FINDXY *)
2970   1   23:0    94
2971   1   24:D     1 PROCEDURE SHOWCURSOR;
2972   1   24:D     1 VAR
2973   1   24:D     1   X,Y: INTEGER;
2974   1   24:0     0 BEGIN
2975   1   24:1     0   FINDXY(X,Y);
2976   1   24:1     6   GOTOXY(X,Y)
2977   1   24:0    11 END(* SHOWCURSOR *);
2978   1   24:0    24
2979   1   25:D     3 FUNCTION GETNUM(*:INTEGER*);
2980   1   25:D     3 VAR
```

```
2981   1   25:D    3   N: INTEGER;
2982   1   25:D    4   OVERFLOW: BOOLEAN;
2983   1   25:0    0 BEGIN
2984   1   25:1    0   N:=0;
2985   1   25:1    3   OVERFLOW:=FALSE;
2986   1   25:1    6   IF NOT (CH IN ['0'..'9']) THEN N:=1
2987   1   25:1   23   ELSE
2988   1   25:2   28     REPEAT
2989   1   25:3   28       IF N > 1000 THEN OVERFLOW:=TRUE
2990   1   25:3   35       ELSE
2991   1   25:4   40         BEGIN
2992   1   25:5   40           N:=N*10+ORD(CH)-ORD('0');
2993   1   25:5   49           CH:=GETCH
2994   1   25:4   49         END
2995   1   25:2   55     UNTIL (NOT (CH IN ['0'..'9'])) OR OVERFLOW;
2996   1   25:1   73   IF OVERFLOW THEN
2997   1   25:2   76     BEGIN
2998   1   25:3   76       ERROR('REPEATFACTOR > 10,000',NONFATAL);
2999   1   25:3   03       GETNUM:=0;
3000   1   25:2   06     END
3001   1   25:1   06   ELSE
3002   1   25:2   08     GETNUM:=N;
3003   1   25:1   11   COMMAND:=MAPTOCOMMAND(CH); (* TAKES CH AND MAPS IT TO A COMMAND *)
3004   1   25:0   18 END;
3005   1   25:0   32
3006   1   26:D    1 PROCEDURE GETLEADING;
3007   1   26:0    0 BEGIN
3008   1   26:0    0   (* SETS:
3009   1   26:0    0         LINESTART ......... A POINTER TO THE BEGINNING OF THE LINE
3010   1   26:0    0         STUFFSTART ........ A POINTER TO THE BEGINNING OF THE TEXT ON THE LINE
3011   1   26:0    0         BYTES ............. THE NUMBER OF BYTES BETWEEN LINESTART AND
3012   1   26:0    0                             STUFFSTART
3013   1   26:0    0         BLANKS ............ THE INDENTATION OF THE LINE    *)
3014   1   26:1    0   LINESTART:=CURSOR;
3015   1   26:1    3   IF EBUF^[LINESTART]=CHR(EOL) THEN LINESTART:=LINESTART-1; (* FOR SCAN! *)
3016   1   26:1   16   LINESTART:=SCAN(-MAXCHAR,=CHR(EOL),EBUF^[LINESTART])+LINESTART+1;
3017   1   26:1   34   BLANKS:=LEADBLANKS(LINESTART,BYTES);
3018   1   26:1   43   STUFFSTART:=LINESTART+BYTES
3019   1   26:0   44 END (* GETLEADING *);
3020   1   26:0   60
3021   1   27:D    3 FUNCTION OKTODEL (* (CURSOR,ANCHOR: PTRTYPE):BOOLEAN *) ;
```

```
3022   1   27:0     0 BEGIN
3023   1   27:1     0   IF ABS(CURSOR-ANCHOR)>(BUFSIZE-BUFCOUNT)+10 THEN
3024   1   27:2    12     BEGIN
3025   1   27:3    12       MSG:=
3026   1   27:3    15 'THERE IS NO ROOM TO COPY THE DELETION.  DO YOU WISH TO DELETE ANYWAY? (Y/N)';
3027   1   27:3    95       PUTMSG;
3028   1   27:3    97       IF UCLC(GETCH)='Y' THEN OKTODEL:=TRUE ELSE OKTODEL:=FALSE;
3029   1   27:2    17     END
3030   1   27:1    17   ELSE
3031   1   27:2    19     BEGIN
3032   1   27:2    19       (* COPYLINE IS SET BY THE CALLER *)
3033   1   27:3    19       COPYOK:=TRUE; COPYLENGTH:=ABS(CURSOR-ANCHOR);
3034   1   27:3    30       COPYSTART:=BUFSIZE-COPYLENGTH+1;
3035   1   27:3    40       MOVELEFT(EBUF^[MIN(CURSOR,ANCHOR)],EBUF^[COPYSTART],COPYLENGTH);
3036   1   27:3    58       OKTODEL:=TRUE
3037   1   27:2    58     END;
3038   1   27:0    61 END;
3039   1   27:0    74
3040   1   27:0    74
3041   1   28:D     1 PROCEDURE LINEOUT(*VAR PTR:PTRTYPE; BYTES,BLANKS,LINE:INTEGER*);
3042   1   28:D     5 (* WRITE A LINE OUT *)
3043   1   28:D     5 VAR
3044   1   28:D     5   LINEDIST,EOLDIST: INTEGER;
3045   1   28:D     7   T: PACKED ARRAY [0..MAXSW] OF CHAR;
3046   1   28:0     0 BEGIN
3047   1   28:1     0   GOTOXY(BLANKS,LINE);
3048   1   28:1     5   PTR:=PTR+BYTES;
3049   1   28:1    11   EOLDIST:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR]);
3050   1   28:1    25   LINEDIST:=MAX(0,MIN(EOLDIST,SCREENWIDTH-BLANKS+1));
3051   1   28:1    42   MOVELEFT(EBUF^[PTR],T[0],LINEDIST);
3052   1   28:1    52   IF EBUF^[PTR+LINEDIST]<>CHR(EOL) THEN (* LINE TRUNCATION *)
3053   1   28:2    63     BEGIN
3054   1   28:3    63       LINEDIST:=MAX(LINEDIST,1);
3055   1   28:3    71       T[LINEDIST-1]:='!';
3056   1   28:2    78     END;
3057   1   28:1    78   WRITE(T:LINEDIST);
3058   1   28:1    88   PTR:=PTR+EOLDIST+1
3059   1   28:0    93 END;
3060   1   28:0    08
3061   1   29:D     1 PROCEDURE UPSCREEN(*FIRSTLINE,WHOLESCREEN: BOOLEAN; LINE: INTEGER*);
3062   1   29:D     4 (* ZAP, INSERT AND DELETE CALL THIS PROCEDURE TO UPDATE (POSSIBLY PARTIALLY)
```

```
3063   1   29:0     4    THE SCREEN.  FIRSTLINE MEANS ONLY THE LINE THAT THE CURSOR IS ON NEED
3064   1   29:0     4    BE UPDATED.  WHOLESCREEN MEANS THAT EVERYTHING MUST BE UPDATED.  IF
3065   1   29:0     4    NEITHER OF THESE IS TRUE THEN ONLY THE PART OF THE SCREEN THAT'S AFTER
3066   1   29:0     4    THE CURSOR IS UPDATED *)
3067   1   29:0     4 VAR
3068   1   29:0     4   PTR: PTRTYPE;
3069   1   29:0     5
3070   1   29:0     0 BEGIN (* UPSCREEN *)
3071   1   29:1     0   IF FIRSTLINE THEN
3072   1   29:2     3     BEGIN
3073   1   29:3     3       GETLEADING;
3074   1   29:3     5       GOTOXY(0,LINE); ERASETOEOL(0,LINE); (* CLEAN THE LINE *)
3075   1   29:3    14       LINEOUT(LINESTART,BYTES,BLANKS,LINE) (* JUST THIS LINE *)
3076   1   29:2    19     END
3077   1   29:1    21   ELSE
3078   1   29:2    23     IF WHOLESCREEN THEN
3079   1   29:3    26       CENTERCURSOR(TRASH,MIDDLE,TRUE)
3080   1   29:2    33     ELSE (* ONLY UPDATE THE PART OF THE SCREEN AFTER THE CURSOR *)
3081   1   29:3    37       BEGIN
3082   1   29:4    37         GOTOXY(0,LINE); ERASEOS(0,LINE);
3083   1   29:4    46         GETLEADING;
3084   1   29:4    48         PTR:=LINESTART;
3085   1   29:4    51         REPEAT
3086   1   29:5    51           BLANKS:=MIN(LEADBLANKS(PTR,BYTES),SCREENWIDTH);
3087   1   29:5    65           LINEOUT(PTR,BYTES,BLANKS,LINE); (* WRITES OUT THE LINE AT PTR *)
3088   1   29:5    72           LINE:=LINE+1
3089   1   29:4    73         UNTIL (LINE>SCREENHEIGHT) OR (PTR>=BUFCOUNT)
3090   1   29:3    83       END;
3091   1   29:0    86 END;
3092   1   29:0    00
3093   1   30:0     1 PROCEDURE READJUST(*CURSOR:PTRTYPE; DELTA: INTEGER*);
3094   1   30:0     3 (* IF DELTA<0 THEN MOVE ALL AFFECTED MARKERS TO CURSOR.  ALSO ADJUST ALL
3095   1   30:0     3    MARKERS >= CURSOR BY DELTA *)
3096   1   30:0     3 VAR
3097   1   30:0     3   I: INTEGER;
3098   1   30:0     0 BEGIN
3099   1   30:1     0   WITH PAGEZERO DO
3100   1   30:2     0     FOR I:=0 TO COUNT-1 DO
3101   1   30:3    15       IF POFFSET[I]>=CURSOR THEN POFFSET[I]:=MAX(POFFSET[I]+DELTA,CURSOR);
3102   1   30:1    54   IF (COPYSTART>=CURSOR) AND (COPYSTART<BUFCOUNT) THEN
3103   1   30:2    67     COPYSTART:=MAX(COPYSTART+DELTA,CURSOR);
```

```
3104   1   30:0    80 END;
3105   1   30:0    94
3106   1   31:0     1 PROCEDURE THEFIXER(*PARAPTR:PTRTYPE;RFAC:INTEGER;WHOLE:BOOLEAN*);
3107   1   31:0     4 (* PARAPTR POINTS SOMEWHERE IN A PARAGRAPH.  IF WHOLE IS TRUE THEN THE
3108   1   31:0     4    ENTIRE PARAGRAPH IS FILLED, OTHERWISE ONLY THAT DIRECTLY AFTER THE CURSOR
3109   1   31:0     4    IS FILLED.  RFAC, WHEN IMPLEMENTED WILL TELL HOW MANY PARAGRAPHS TO BE
3110   1   31:0     4    FILLED.  NOTE: A PARAGRAPH IS DEFINED AS LINES OF TEXT DELIMITED BY A LINE
3111   1   31:0     4    WITH NO TEXT ON IT WHATSOEVER, OR A LINE OF A TEXT WHOSE FIRST CHARACTER IS
3112   1   31:0     4    RUNOFFCH *)
3113   1   31:0     4
3114   1   31:0     4 VAR
3115   1   31:0     4   SAVE,PTR,WPTR: INTEGER;
3116   1   31:0     7   WLENGTH,X: INTEGER;
3117   1   31:0     9   DONE: BOOLEAN;
3118   1   31:0     0 BEGIN
3119   1   31:1     0   WITH PAGEZERO DO
3120   1   31:2     0     BEGIN
3121   1   31:3     0       SAVE:=CURSOR;
3122   1   31:3     3       CURSOR:=PARAPTR;
3123   1   31:3     6       GETLEADING;
3124   1   31:3     8       IF EBUF^[STUFFSTART] IN [CHR(EOL),RUNOFFCH] THEN EXIT(THEFIXER);
3125   1   31:3    26       IF WHOLE THEN (* SCAN BACKWARDS FOR THE BEGINNING OF THE PARAGRAPH *)
3126   1   31:4    29         BEGIN
3127   1   31:5    29           REPEAT
3128   1   31:6    29             CURSOR:=LINESTART-1;
3129   1   31:6    34             GETLEADING
3130   1   31:5    34           UNTIL (LINESTART<=1) OR (EBUF^[STUFFSTART] IN [RUNOFFCH,CHR(EOL)]);
3131   1   31:5    54           IF EBUF^[STUFFSTART] IN [RUNOFFCH,CHR(EOL)] THEN
3132   1   31:6    68             PTR:=CURSOR+1
3133   1   31:5    69           ELSE
3134   1   31:6    75             PTR:=1;
3135   1   31:5    78           X:=PARAMARGIN;
3136   1   31:4    83         END
3137   1   31:3    83       ELSE
3138   1   31:4    85         BEGIN
3139   1   31:5    85           PTR:=LINESTART;
3140   1   31:5    88           IF BLANKS=PARAMARGIN THEN X:=PARAMARGIN ELSE X:=LMARGIN
3141   1   31:4    02         END;
3142   1   31:3    07       CURSOR:=BUFSIZE-(BUFCOUNT-PTR)+1; (* SPLIT THE BUFFER *)
3143   1   31:3    16       MOVERIGHT(EBUF^[PTR],EBUF^[CURSOR],BUFCOUNT-PTR);
3144   1   31:3    27       (* NOW DRIBBLE BACK THE (REST OF THE) PARAGRAPH *)
```

```
3145   1   31:3   27        EBUF^[PTR]:=CHR(DLE);
3146   1   31:3   32        EBUF^[PTR+1]:=CHR(X+32);
3147   1   31:3   41        PTR:=PTR+2;
3148   1   31:3   46        EBUF^[CURSOR-1]:=CHR(EOL); (* SENTINEL FOR GETLEADING *)
3149   1   31:3   53        DONE:=FALSE;
3150   1   31:3   56        REPEAT
3151   1   31:4   56          WHILE EBUF^[CURSOR] IN [CHR(HT),CHR(SP),CHR(DLE)] DO
3152   1   31:5   71            IF EBUF^[CURSOR]=CHR(DLE) THEN CURSOR:=CURSOR+2 ELSE CURSOR:=CURSOR+
3153   1   31:4   93          WPTR:=CURSOR;
3154   1   31:4   96          (* SKIP OVER A TOKEN *)
3155   1   31:4   96          WHILE NOT (EBUF^[CURSOR] IN [CHR(EOL),' ','-']) DO CURSOR:=CURSOR+1;
3156   1   31:4   23          (* SPECIAL CASES FOR ".<SP><SP>" AND "-<SP>" *)
3157   1   31:4   23          IF EBUF^[CURSOR]='-' THEN IF EBUF^[CURSOR+1]=' ' THEN CURSOR:=CURSOR+1
3158   1   31:4   46          IF (EBUF^[CURSOR-1]='.') THEN IF
3159   1   31:5   56             (EBUF^[CURSOR]=' ') AND (EBUF^[CURSOR+1]=' ') THEN CURSOR:=CURSOR+1
3160   1   31:4   78          WLENGTH:=CURSOR-WPTR+1; (* INCLUDING THE DELIMITER *)
3161   1   31:4   85          IF (X+WLENGTH>RMARGIN) OR (RMARGIN-LMARGIN+1<=WLENGTH) THEN
3162   1   31:5   06            BEGIN
3163   1   31:6   06              IF EBUF^[PTR-1]=' ' THEN PTR:=PTR-1;
3164   1   31:6   21              EBUF^[PTR]:=CHR(EOL); EBUF^[PTR+1]:=CHR(DLE);
3165   1   31:6   33              EBUF^[PTR+2]:=CHR(LMARGIN+32);
3166   1   31:6   44              PTR:=PTR+3;
3167   1   31:6   49              X:=LMARGIN
3168   1   31:5   49            END;
3169   1   31:4   54          CURSOR:=CURSOR+1;
3170   1   31:4   59          MOVELEFT(EBUF^[WPTR],EBUF^[PTR],WLENGTH);
3171   1   31:4   68          IF EBUF^[CURSOR-1]=CHR(EOL) THEN
3172   1   31:5   78            BEGIN
3173   1   31:6   78              IF EBUF^[CURSOR]=CHR(0) THEN DONE:=TRUE
3174   1   31:6   86              ELSE
3175   1   31:7   91                BEGIN
3176   1   31:8   91                  GETLEADING;
3177   1   31:8   93                  DONE:=(EBUF^[STUFFSTART]=CHR(EOL))
3178   1   31:8   99                        OR (EBUF^[STUFFSTART]=RUNOFFCH);
3179   1   31:8   10                  (* THE LAST TRANSFER WILL MOVE
3180   1   31:8   10                     OVER THE <EOL> FOR THE PARAGRAPH *)
3181   1   31:8   10                  IF NOT DONE THEN
3182   1   31:9   14                    BEGIN
3183   1   31:0   14                      EBUF^[PTR+WLENGTH-1]:=' ';
3184   1   31:0   23                      (* IF <EOL> <SP>, MAP TO ONE SPACE ONLY *)
3185   1   31:0   23                      IF EBUF^[CURSOR-2]=' ' THEN PTR:=PTR-1;
```

```
3186  1   31:9   38                    END
3187  1   31:7   38                  END
3188  1   31:5   38                END;
3189  1   31:4   38              X:=X+WLENGTH;
3190  1   31:4   43              PTR:=PTR+WLENGTH;
3191  1   31:3   48            UNTIL DONE;
3192  1   31:3   51            READJUST(PARAPTR,(BUFSIZE-CURSOR+PTR+1)-BUFCOUNT);
3193  1   31:3   63            BUFCOUNT:=BUFSIZE-CURSOR+PTR+1;
3194  1   31:3   72            MOVELEFT(EBUF^[CURSOR],EBUF^[PTR],BUFSIZE-CURSOR+1);
3195  1   31:3   85            EBUF^[BUFCOUNT]:=CHR(0);
3196  1   31:3   90            CURSOR:=MIN(BUFCOUNT-1,SAVE);
3197  1   31:3   00            GETLEADING;
3198  1   31:3   02            CURSOR:=MAX(CURSOR,STUFFSTART)
3199  1   31:2   04        END;
3200  1   31:0   10    END;
3201  1   31:0   30
3202  1   32:0    1    PROCEDURE GETNAME(*MSG:STRING; VAR M:NAME*);
3203  1   32:0   44    VAR
3204  1   32:0   44      I: INTEGER;
3205  1   32:0   45      S: STRING;
3206  1   32:0    0    BEGIN
3207  1   32:1    0      NEEDPROMPT:=TRUE; HOME; CLEARLINE(0); WRITE(MSG,' WHAT MARKER? ');
3208  1   32:1   47      READLN(S);
3209  1   32:1   62      FOR I:=1 TO LENGTH(S) DO S[I]:=UCLC(S[I]);
3210  1   32:1   00      MOVELEFT(S[1],M[0],MIN(8,LENGTH(S)));
3211  1   32:1   16      FILLCHAR(M[LENGTH(S)],MAX(0,8-LENGTH(S)),' ')
3212  1   32:0   35    END;
3213  1   32:0   50
3214  1   32:0   50
3215  1   32:0   50    (*$I UTIL      *)
3216  1   32:0   50
3217  1    1:0    0    BEGIN (* SEGMENT PROCEDURE EDITOR *)
3218  1    1:1    0      INITIALIZE; GETLEADING; CURSOR:=MAX(CURSOR,STUFFSTART);
3219  1    1:1   26      REPEAT
3220  1    1:2   26        CENTERCURSOR(TRASH,(SCREENHEIGHT DIV 2)+1,TRUE);
3221  1    1:2   37        NEEDPROMPT:=TRUE;
3222  1    1:2   41        IF USERINFO.ERRBLK>0 THEN PUTSYNTAX;
3223  1    1:2   51        REPEAT
3224  1    1:3   51          HOME; CLEARLINE(0);
3225  1    1:3   56          EDITCORE;
3226  1    1:3   59          IF COMMAND=SETC THEN ENVIRONMENT
```

```
3227    1     1:3     64        ELSE IF COMMAND=COPYC THEN COPYFILE
3228    1     1:2     74      UNTIL COMMAND=QUITC;
3229    1     1:1     82    UNTIL OUT;
3230    1     1:1     89    SYSCOM^.MISCINFO.NOBREAK := FALSE  (* 28 SEPT 77*)
3231    1     1:0     96 END;
3232    1     1:0     22
3233    0     1:0      0 BEGIN END.
```

```
 2   1    1:0    1 (*$L PRINTER: *)
 3   1    1:0    1 (*$S+*)
 4   1    1:0    1
 5   1    1:0    1 (*************************************************************************)
 6   1    1:0    1 (*                                                                       *)
 7   1    1:0    1 (*     SCREEN ORIENTED EDITOR                          JULY 8, 1978      *)
 8   1    1:0    1 (*     ------ -------- ------                                            *)
 9   1    1:0    1 (*                                                      ---------        *)
10   1    1:0    1 (*     BY RICHARD S. KAUFMANN,                         /         \       *)
11   1    1:0    1 (*        IIS                                          \ VERSION \       *)
12   1    1:0    1 (*        UNIVERSITY OF CALIFORNIA, SAN DIEGO          \   L.2   \       *)
13   1    1:0    1 (*        LA JOLLA CA 92093                            \---------/       *)
14   1    1:0    1 (*                                                                       *)
15   1    1:0    1 (*     COPYRIGHT (C) 1978, BY THE REGENTS OF THE UNIVERSITY OF           *)
16   1    1:0    1 (*        CALIFORNIA AT SAN DIEGO                                        *)
17   1    1:0    1 (*                                                                       *)
18   1    1:0    1 (*************************************************************************)
19   1    1:0    1
20   1    1:0    1 (*$U-*)
21   0    1:0    1 PROGRAM PASCALSYSTEM;
22   0    1:0    1
23   0    1:0    1 CONST
24   0    1:0    1   VIDLENG = 7;  (* NUMBER OF CHARACTERS IN A VOLUME ID *)
25   0    1:0    1   TIDLENG = 15; (* NUMBER OF CHARACTERS IN A TITLE ID *)
26   0    1:0    1
27   0    1:0    1 TYPE
28   0    1:0    1
29   0    1:0    1   VID = STRING[VIDLENG];
30   0    1:0    1
31   0    1:0    1   TID = STRING[TIDLENG];
32   0    1:0    1
33   0    1:0    1   DATEREC=PACKED RECORD
34   0    1:0    1              MONTH: 0..12;
35   0    1:0    1              DAY:   0..31;
36   0    1:0    1              YEAR:  0..100
37   0    1:0    1            END;
38   0    1:0    1
39   0    1:0    1   INFOREC = RECORD
40   0    1:0    1               TRASH1,TRASH2: INTEGER;
41   0    1:0    1               ERRSYM,ERRBLK,ERRNUM: INTEGER;    (* ERROR COM FOR EDIT *)
42   0    1:0    1               TRASH3: ARRAY [0..2] OF INTEGER;
```

```
 43    0     1:0      1                GOTSYM,GOTCODE: BOOLEAN;
 44    0     1:0      1                WORKVID,SYMVID,CODEVID: VID;        (* PERM&CUR WORKFILE VOLUMES *)
 45    0     1:0      1                WORKTID,SYMTID,CODETID: TID         (* PERM&CUR WORKFILE TITLES *)
 46    0     1:0      1              END (*INFOREC*) ;
 47    0     1:0      1
 48    0     1:0      1   SYSCOMREC = RECORD
 49    0     1:0      1                 JUNK: ARRAY [0..6] OF INTEGER;
 50    0     1:0      1                 LASTMP: INTEGER;
 51    0     1:0      1                 EXPANSION: ARRAY [0..20] OF INTEGER;
 52    0     1:0      1                 MISCINFO: PACKED RECORD
 53    0     1:0      1                             NOBREAK,STUPID,SLOWTERM,
 54    0     1:0      1                             HASXYCRT,HASLCCRT,HAS8510A,HASCLOCK: BOOLEAN
 55    0     1:0      1                           END;
 56    0     1:0      1                 CRTTYPE: INTEGER;
 57    0     1:0      1                 CRTCTRL: PACKED RECORD
 58    0     1:0      1                             RLF,NDFS,ERASEEOL,ERASEEOS,HOME,ESCAPE: CHAR;
 59    0     1:0      1                             BACKSPACE: CHAR;
 60    0     1:0      1                             FILLCOUNT: 0..255;
 61    0     1:0      1                             EXPANSION: PACKED ARRAY [0..3] OF CHAR
 62    0     1:0      1                           END;
 63    0     1:0      1                 CRTINFO: PACKED RECORD
 64    0     1:0      1                             WIDTH,HEIGHT: INTEGER;
 65    0     1:0      1                             RIGHT,LEFT,DOWN,UP: CHAR;
 66    0     1:0      1                             BADCH,CHARDEL,STOP,BREAK,FLUSH,EOF: CHAR;
 67    0     1:0      1                             ALTMODE,LINEDEL: CHAR;
 68    0     1:0      1                             EXPANSION: PACKED ARRAY [0..5] OF CHAR
 69    0     1:0      1                           END
 70    0     1:0      1              END (*SYSCOM*);
 71    0     1:0      1
 72    0     1:0      1 VAR (* I.4 GLOBALS AS OF 30-JAN-78 *)
 73    0     1:0      1   SYSCOM: ^SYSCOMREC;
 74    0     1:0      2   TRASHY: ARRAY [0..5] OF INTEGER;
 75    0     1:0      8   USERINFO: INFOREC;
 76    0     1:0     54   TRASHYY: ARRAY [0..4] OF INTEGER;
 77    0     1:0     59   SYVID,DKVID: VID;
 78    0     1:0     67   THEDATE: DATEREC;
 79    0     1:0     68
 80    0     1:0     68
 81    0     1:0     68 (*$TEDITOR SEGMENT*)
 82    1     1:0      1 SEGMENT PROCEDURE EDITOR;
 83    1     1:0      1 CONST
```

```
 34    1     1:0      1    (* UNLESS OTHERWISE NOTED ALL CONSTANTS ARE UPPER BOUNDS
 35    1     1:0      1       FROM ZERO.                                          *)
 36    1     1:0      1
 37    1     1:0      1    MAXBUFSIZE=32767;
 38    1     1:0      1    MAXSW=34; (* MAXIMUM ALLOWABLE SCREENWIDTH *)
 39    1     1:0      1    MAXSTRING=127;
 90    1     1:0      1    MAXCHAR=1023;  (* THE MAXIMUM NUMBER OF CHARACTERS ON A LINE IN THE EBUF *)
 91    1     1:0      1    TIDLENG=15;  (* FROM SYSCOM *)
 92    1     1:0      1    CHARINBUF=2048;  (* FOR FINAL VERSION. NOT USED. *)
 93    1     1:0      1    MAXOFFSET=1023; (* MAXIMUM OFFSET IN A PAGE *)
 94    1     1:0      1    MAXPAGE=255; (* RIDICULOUS UPPER BOUND! *)
 95    1     1:0      1
 96    1     1:0      1    (* THE FOLLOWING ASCII CHARACTERS ARE HARD-WIRED IN *)
 97    1     1:0      1    BSPCE=8; HT=9; LF=10; EOL=13; DLE=16; SP=32;
 98    1     1:0      1    DC1=17; BELL=7; RUBOUT=127; CR=13;
 99    1     1:0      1
100    1     1:0      1
101    1     1:0      1  TYPE
102    1     1:0      1    PTRTYPE=0..MAXBUFSIZE;
103    1     1:0      1    BUFRTYPE=PACKED ARRAY [0..0] OF CHAR;
104    1     1:0      1    BLOCKTYPE=PACKED ARRAY [0..511] OF CHAR;
105    1     1:0      1    ERRORTYPE=(FATAL,NONFATAL);
106    1     1:0      1    TABATTRIBUTE=(NONE,LEFTJUST,RIGHTJUST,DECIMALSTOP);
107    1     1:0      1    OFFSET=0..MAXOFFSET;
108    1     1:0      1    PAGE=0..MAXPAGE;
109    1     1:0      1    NAME=PACKED ARRAY [0..7] OF CHAR;
110    1     1:0      1    PTYPE=PACKED ARRAY [0..MAXSTRING] OF CHAR;
111    1     1:0      1    COMMANDS=(ILLEGAL, ADJUSTC, BANISHC, COPYC, DELETEC, FINDC, INSERTC, JUMPC,
112    1     1:0      1              LISTC, MACRODEFC, NEXTC, PARAC, QUITC, REPLACEC, SETC, VERIFYC,
113    1     1:0      1              XECUTEC, ZAPC, REVERSEC, FORWARDC, UP, DOWN, LEFT, RIGHT, TAB,
114    1     1:0      1              DIGIT, DUMPC, ADVANCE, SPACE, EQUALC, SLASHC);
115    1     1:0      1    CTYPE=(FS,GOHOME,ETOEOL,ETOEOS,US);
116    1     1:0      1    LEFTRIGHT=(LEFTSTACK,RIGHTSTACK);
117    1     1:0      1
118    1     1:0      1    HEADER=     (* PAGE ZERO LAYOUT CHANGED 20-JUN-78 *)
119    1     1:0      1      RECORD CASE BOOLEAN OF
120    1     1:0      1        TRUE: (BUF: PACKED ARRAY[0..MAXOFFSET] OF CHAR);
121    1     1:0      1        FALSE:(DEFINED:    INTEGER; (* NEW FILE => 0; OLD FILE => 1 *)
122    1     1:0      1               COUNT:      INTEGER;  (* THE COUNT OF VALID MARKERS *)
123    1     1:0      1               NAME:       ARRAY [0..19] OF PACKED ARRAY [0..7] OF CHAR;
124    1     1:0      1               PAGEN:      PACKED ARRAY [0..19] OF INTEGER;
```

```
125   1     1:D     1                POFFSET:     PACKED ARRAY [0..19] OF OFFSET;
126   1     1:D     1                TABSTOP:     PACKED ARRAY [0..127] OF TABATTRIBUTE;
127   1     1:D     1                AUTOINDENT:  BOOLEAN; (* ENVIRONMENT STUFF FOLLOWS *)
128   1     1:D     1                FILLING:     BOOLEAN;
129   1     1:D     1                TOKDEF:      BOOLEAN;
130   1     1:D     1                LMARGIN:     0..MAXSw;
131   1     1:D     1                RMARGIN:     0..MAXSw;
132   1     1:D     1                PARAMARGIN:  0..MAXSw;
133   1     1:D     1                RUNOFFCH:    CHAR;
134   1     1:D     1                CREATED:     DATEREC;
135   1     1:D     1                LASTUPD:     DATEREC;
136   1     1:D     1                REVISION:    INTEGER;
137   1     1:D     1                FILLER:      ARRAY [0..91] OF INTEGER)
138   1     1:D     1      END;
139   1     1:D     1
140   1     1:D     1
141   1     1:D     1
142   1     1:D     1  VAR
143   1     1:D     1    CURSOR: 0..MAXBUFSIZE;
144   1     1:D     2    BUFCOUNT: 0..MAXBUFSIZE;          (* NUMBER OF VALID CHARACTERS IN THE EBUF *)
145   1     1:D     3    STUFFSTART: 0..MAXBUFSIZE;        (* GETLEADING              *)
146   1     1:D     4    LINESTART: 0..MAXBUFSIZE;         (*              SETS       *)
147   1     1:D     5    BYTES,BLANKS: INTEGER;            (*                   THESE *)
148   1     1:D     7    CH: CHAR;
149   1     1:D     8    DIRECTION: CHAR;                  (* '>' OR '<' *)
150   1     1:D     9    REPEATFACTOR: INTEGER;
151   1     1:D    10    BUFSIZE: INTEGER;
152   1     1:D    11    SCREENWIDTH: INTEGER;             (* MOVED TO VAR 26-JAN *)
153   1     1:D    12    SCREENHEIGHT: INTEGER;            (*   "    "  "    "    *)
154   1     1:D    13    COMMAND: COMMANDS;
155   1     1:D    14    LASTPAT: 0..MAXBUFSIZE;
156   1     1:D    15    EBUF: ^BUFRTYPE;
157   1     1:D    16    KIND: ARRAY [CHAR] OF INTEGER;    (* FOR TOKEN FIND *)
158   1     1:D   272    LINE1PTR: 0..MAXBUFSIZE;
159   1     1:D   273    MIDDLE: INTEGER;                  (* MIDDLE LINE ON THE SCREEN *)
160   1     1:D   274    NEEDPROMPT: BOOLEAN;
161   1     1:D   275    ETX,BS,DEL,ESC: INTEGER;          (* MOVED FROM CONST 30-JAN-78 *)
162   1     1:D   279    FLENGTH: INTEGER;                 (* THE LENGTH OF THE WORKFILE IN PAGES *)
163   1     1:D   280    LPAGE,RPAGE: INTEGER;             (* LEFT AND RIGHT STACK POINTERS *)
164   1     1:D   282    TRASH: INTEGER;                   (* TOTALLY WITHOUT REDEEMING SOCIAL VALUE *)
165   1     1:D   283    TARGET: PTYPE;
```

```
166   1     1:0    347   SUBSTRING: PTYPE;
167   1     1:0    411   SLENGTH,TLENGTH: INTEGER;          (* LENGTH OF TARGET AND SUBSTRING *)
168   1     1:0    413   SDEFINED,TDEFINED: BOOLEAN;        (* WHETHER THE STRINGS ARE VALID *)
169   1     1:0    415   COPYLENGTH,COPYSTART: PTRTYPE;     (* FOR COPYC *)
170   1     1:0    417   COPYLINE,COPYOK: BOOLEAN;          (*     "      *)
171   1     1:0    419   INFINITY: BOOLEAN;                 (* FOR SLASHC *)
172   1     1:0    420   THEFILE: FILE;
173   1     1:0    460   PR: FILE;                          (* DEBUG *)
174   1     1:0    500   TRANSLATE: ARRAY [CHAR] OF COMMANDS;
175   1     1:0    756   PAGEZERO: HEADER;
176   1     1:0   1268   MSG: STRING;
177   1     1:0   1309   PROMPTLINE: STRING;
178   1     1:0   1350   SAVETOP: STRING;                   (* DUMB TERMINAL PATCH - FOR BLANKCRT(1) *)
179   1     1:0   1391   PAGEBUFFER: PACKED ARRAY [0..1023] OF CHAR;
180   1     1:0   1903   BLANKAREA: PACKED ARRAY [0..MAXSW] OF CHAR;
181   1     1:0   1946   WFNAME,BACKFNAME: STRING;
182   1     1:0   2028
183   3     1:5      0 SEGMENT PROCEDURE NUM2; BEGIN END;       SEGMENT PROCEDURE NUM3; BEGIN END;
184   5     1:0      0 SEGMENT PROCEDURE NUM4; BEGIN END;       SEGMENT PROCEDURE NUM5; BEGIN END;
185   7     1:0      0 SEGMENT PROCEDURE NUM6; BEGIN END;       SEGMENT PROCEDURE NUM7; BEGIN END;
186   9     1:0      0 SEGMENT PROCEDURE NUM8; BEGIN END;       SEGMENT PROCEDURE NUM9; BEGIN END;
187   9     1:0     12
188   9     1:0     12
189   9     1:0     12
190   9     1:0     12
191   9     1:0     12 (* FORWARD DECLARED PROCEDURES.. ALL PROCEDURES ARE IN MISC AND UTIL *)
192   9     1:0     12
193   1     2:0      1 PROCEDURE ERROR(S:STRING;HOWBAD:ERRORTYPE); FORWARD;
194   1     3:0      1 PROCEDURE ERASETOEOL(X,LINE:INTEGER); FORWARD;
195   1     4:0      3 FUNCTION  GETCH:CHAR; FORWARD;
196   1     5:0      1 PROCEDURE CLEARSCREEN; FORWARD;
197   1     6:0      1 PROCEDURE ERASEOS(X,LINE:INTEGER); FORWARD;
198   1     7:0      1 PROCEDURE CLEARLINE(Y:INTEGER); FORWARD;
199   1     8:0      3 FUNCTION  MAPTOCOMMAND(CH:CHAR): COMMANDS; FORWARD;
200   1     9:0      3 FUNCTION  UCLC(CH:CHAR): CHAR; FORWARD;
201   1    10:0      1 PROCEDURE PROMPT; FORWARD;
202   1    11:0      1 PROCEDURE REDISPLAY; FORWARD;
203   1    12:0      3 FUNCTION  MIN(A,B:INTEGER): INTEGER; FORWARD;
204   1    13:0      3 FUNCTION  MAX(A,B:INTEGER): INTEGER; FORWARD;
205   1    14:0      1 PROCEDURE CONTROL(CH:CTYPE); FORWARD;
206   1    15:0      1 PROCEDURE PUTMSG; FORWARD;
```

```
207   1   16:0      1 PROCEDURE HOME; FORWARD;
208   1   17:0      1 PROCEDURE ERRWAIT; FORWARD;
209   1   18:0      1 PROCEDURE BLANKCRT(Y: INTEGER); FORWARD;
210   1   19:0      3 FUNCTION  LEADBLANKS(PTR:PTRTYPE;VAR BYTES: INTEGER): INTEGER; FORWARD;
211   1   20:0      1 PROCEDURE CENTERCURSOR(VAR LINE: INTEGER; LINESUP: INTEGER; NEWSCREEN:BOOLEAN);
212   1   20:0      4   FORWARD;
213   1   21:0      1 PROCEDURE FINDXY(VAR INDENT,LINE: INTEGER); FORWARD;
214   1   22:0      1 PROCEDURE SHOWCURSOR; FORWARD;
215   1   23:0      3 FUNCTION  GETNUM: INTEGER; FORWARD;
216   1   24:0      1 PROCEDURE GETLEADING; FORWARD;
217   1   25:0      3 FUNCTION  OKTODEL(CURSOR,ANCHOR:PTRTYPE):BOOLEAN; FORWARD;
218   1   26:0      1 PROCEDURE LINEOUT(VAR PTR:PTRTYPE; BYTES,BLANKS,LINE: INTEGER); FORWARD;
219   1   27:0      1 PROCEDURE UPSCREEN(FIRSTLINE,WHOLESCREEN:BOOLEAN; LINE: INTEGER); FORWARD;
220   1   28:0      1 PROCEDURE READJUST(CURSOR: PTRTYPE; DELTA: INTEGER); FORWARD;
221   1   29:0      1 PROCEDURE THEFIXER(PARAPTR: PTRTYPE;RFAC: INTEGER;WHOLE: BOOLEAN); FORWARD;
222   1   30:0      1 PROCEDURE GETNAME(MSG:STRING; VAR M:NAME); FORWARD;
223   1   31:0      1 PROCEDURE GETPAGES(WHICH:LEFTRIGHT); FORWARD;
224   1   32:0      1 PROCEDURE PUTPAGES(WHICH:LEFTRIGHT); FORWARD;
225   1   33:0      3 FUNCTION  READIT(WHICH:LEFTRIGHT): BOOLEAN; FORWARD;
226   1   34:0      3 FUNCTION  WRITEIT(WHICH:LEFTRIGHT): BOOLEAN; FORWARD;
227   1   35:0      1 PROCEDURE CHECKINDENT(VAR CURSOR:PTRTYPE); FORWARD;
228   1   35:0      2
229   1   35:0      2 (*$TI N I T I A L I Z E*)
230  10    1:0      1 SEGMENT PROCEDURE INITIALIZE;
231  10    1:0      1 LABEL 1;
232  10    1:0      1 TYPE PHYLE=FILE;
233  10    1:0      1 VAR
234  10    1:0      1   BLOCK: ^BLOCKTYPE;
235  10    1:0      2   ONEWD: ^INTEGER;
236  10    1:0      3   DONE,OVFLW: BOOLEAN;
237  10    1:0      5   CH: CHAR;
238  10    1:0      6   I,QUIT,GAP,BLKS,PAGE,NOTNULS: INTEGER;
239  10    1:0     12   FILENAME: STRING;
240  10    1:0     53   BUFFER: PACKED ARRAY [0..1023] OF CHAR;
241  10    1:0    565   FIBAREA: ARRAY [0..17] OF INTEGER;
242  10    1:0    583
243  10    2:0      1 PROCEDURE MAP(CH:CHAR; C:COMMANDS);
244  10    2:0      0 BEGIN
245  10    2:1      0   TRANSLATE[CH]:=C;
246  10    2:1      8   IF CH IN ['A'..'Z'] THEN TRANSLATE[CHR(32+ORD(CH))]:=C; (* LC TOO *)
247  10    2:0     38 END;
```

```
248  10    2:0    50
249  10    3:0     1 PROCEDURE CLEANTITLE(VAR T:STRING);
250  10    3:0     2 (* ATTACHES THE DEFAULT '.TEXT' TO THE END OF THE FILENAME IF NECESSARY. *)
251  10    3:0     0 BEGIN
252  10    3:1     0   FOR I:=1 TO LENGTH(T) DO T[I]:=UCLC(T[I]);
253  10    3:1    41   IF (POS('.TEXT',T)=LENGTH(T)-4) AND (LENGTH(T)>=5) THEN
254  10    3:2    69     DELETE(T,LENGTH(T)-4,5);
255  10    3:1    79   WFNAME:=CONCAT(T,'.TEXT');
256  10    3:1   110   BACKFNAME:=CONCAT(T,'.BACK');
257  10    3:0   141 END;
258  10    3:0   156
259  10    4:0     1 PROCEDURE DEFAULTPZ;
260  10    4:0     0 BEGIN
261  10    4:1     0   WITH PAGEZERO DO
262  10    4:2     0     IF DEFINED<>2 THEN
263  10    4:3     7       BEGIN
264  10    4:4     7         FILLCHAR(BUF,1024,CHR(0));
265  10    4:4    17         TOKDEF:=TRUE; (* DEFAULT MODE IS T(OKEN *)
266  10    4:4    21         FILLING:=FALSE; AUTOINDENT:=TRUE; RUNOFFCH:='^';
267  10    4:4    33         LMARGIN:=0; PARAMARGIN:=5; RMARGIN:=SCREENWIDTH;
268  10    4:4    45         (* INITIALIZE TABSTOPS - 20-JUN-78 *)
269  10    4:4    45         FOR I:=0 TO 15 DO TABSTOP[I*8]:=LEFTJUST;
270  10    4:4    82         CREATED:=THEDATE; REVISION:=-1; LASTUPD:=THEDATE;
271  10    4:4   103         DEFINED:=2;
272  10    4:3   107       END;
273  10    4:0   107 END;
274  10    4:0   122
275  10    5:0     1 PROCEDURE CHANGENAME(VAR F:PHYLE; T:STRING);
276  10    5:0    44 (* CHANGE THE TITLE OF F TO T.  NOTE: (1) THE FILE F MUST BE CLOSED WITH
277  10    5:0    44    CLOSE(F,LOCK), AND (2) THIS CODE RELIES ON A "SPECIAL FEATURE" IN THE
278  10    5:0    44    I/O SUBSYSTEM, NAMELY WHEN THE YEAR IS SET TO 100 THE TITLE GETS UPDATED
279  10    5:0    44    WHEN THE FILE IS CLOSED *)
280  10    5:0    44 VAR
281  10    5:0    44   COLON: INTEGER;
282  10    5:0    45   D: DATEREC;
283  10    5:0    46   FIBPA: PACKED ARRAY [0..57] OF CHAR;
284  10    5:0     0 BEGIN
285  10    5:0     0   (* MAKE SURE THAT THE FILENAME DOESN'T INCLUDE THE VOLUME NAME (OR "*") *)
286  10    5:1     0   COLON:=POS(':',T);
287  10    5:1    18   IF COLON>0 THEN DELETE(T,1,COLON);
288  10    5:1    32   IF T[1]='*' THEN DELETE(T,1,1);
```

```
289  10    5:1    47   MOVELEFT(F,FIBPA,58); (* TRANSFERS THE FIB FOR THE FILE F TO FIBPA *)
290  10    5:1    55   MOVELEFT(T,FIBPA[38],16);
291  10    5:1    84   WITH D DO BEGIN DAY:=2; MONTH:=3; YEAR:=100 END;
292  10    5:1    82   MOVELEFT(D,FIBPA[56],2);
293  10    5:1    91   MOVELEFT(FIBPA,F,58)
294  10    5:0    99 END;
295  10    5:0   112
296  10    6:0     3 FUNCTION FINDLENGTH(VAR F:PHYLE):INTEGER;
297  10    6:0     0 BEGIN
298  10    6:0     0   (* KLUDGE LOGIC.  RETURNS THE LENGTH OF THE FILE IN PAGES! *)
299  10    6:1     0   MOVELEFT(F,FIBAREA,36);
300  10    6:1    10   FINDLENGTH:=(FIBAREA[17]-FIBAREA[16]) DIV 2;
301  10    6:0    31 END;
302  10    6:0    44
303  10    7:0     1 PROCEDURE BACKUP;
304  10    7:0     1 (* COPIES THE FILE TO BE EDITED TO ANOTHER FILE, NAMES THE ORIGINAL .BACK, AND
305  10    7:0     1    NAMES THE COPY .TEXT *)
306  10    7:0     1 VAR
307  10    7:0     1   INBNUM,OUTBNUM,OUTFSIZE,BLKSREAD,MAXBLOCKINBUF: INTEGER;
308  10    7:0     6   CH: CHAR;
309  10    7:0     7   F: FILE;
310  10    7:0     0 BEGIN
311  10    7:1     0   REWRITE(F,BACKFNAME);
312  10    7:1    22   IF IORESULT<>0 THEN ERROR('CAN''T OPEN BACKUP FILE!  ',FATAL);
313  10    7:1    60   OUTFSIZE:=FINDLENGTH(F);
314  10    7:1    68   IF OUTFSIZE<FLENGTH THEN ERROR('NOT ENOUGH ROOM FOR BACKUP!  ',FATAL);
315  10    7:1   111   WRITELN('COPYING TO ',BACKFNAME);
316  10    7:1   148   RPAGE:=OUTFSIZE-FLENGTH+1;                    (* PUSH TEXT TO THE RIGHT *)
317  10    7:1   158   INBNUM:=2;                          (* FIRST VALID PAGE IN THE INPUT FILE *)
318  10    7:1   161   OUTBNUM:=RPAGE+RPAGE; (* FIRST BLOCK TO COPY STUFF TO - RIGHT JUSTIFIED *)
319  10    7:1   170   (* COPY OVER THE PAGE ZERO *)
320  10    7:1   170   IF BLOCKREAD(THEFILE,PAGEZERO,2,0)<>2 THEN
321  10    7:2   189     ERROR('READING PAGE ZERO',FATAL);
322  10    7:2   213   (* COMPENSATE FOR SHIFT IN FILE *)
323  10    7:1   213   WITH PAGEZERO DO
324  10    7:2   213     FOR I:=0 TO COUNT-1 DO
325  10    7:3   232       PAGEN[I]:=PAGEN[I]+RPAGE-1;
326  10    7:1   266   IF BLOCKWRITE(F,PAGEZERO,2,0)<>2 THEN
327  10    7:2   284     ERROR('WRITING PAGE ZERO',FATAL);
328  10    7:1   308   MAXBLOCKINBUF:=BUFSIZE DIV 512;
329  10    7:1   315   REPEAT
```

```
330  10    7:2   315      BLKSREAD:=BLOCKREAD(THEFILE,EBUF^,MAXBLOCKINBUF,INBNUM);
331  10    7:2   330      IF IORESULT<>0 THEN ERROR('BAD INPUT FILE.',FATAL);
332  10    7:2   358      IF BLKSREAD<>0 THEN
333  10    7:3   363        BEGIN
334  10    7:4   383          IF BLOCKWRITE(F,EBUF^,BLKSREAD,OUTBNUM)<>BLKSREAD THEN
335  10    7:5   379            ERROR('RAN OUT OF ROOM.',FATAL);
336  10    7:4   402          IF IORESULT<>0 THEN ERROR('ON BACKUP FILE.',FATAL);
337  10    7:3   430        END;
338  10    7:2   430      OUTBNUM:=OUTBNUM+BLKSREAD;
339  10    7:2   435      INBNUM:=INBNUM+BLKSREAD
340  10    7:1   436    UNTIL BLKSREAD=0;
341  10    7:1   445    CHANGENAME(THEFILE,BACKFNAME);
342  10    7:1   453    CLOSE(THEFILE,LOCK);
343  10    7:1   460    CHANGENAME(F,WFNAME);
344  10    7:1   467    CLOSE(F,LOCK);
345  10    7:1   473    FLENGTH:=OUTFSIZE;     (* COPY OVER THE LENGTH ATTRIBUTE,       *)
346  10    7:1   477    RESET(THEFILE,WFNAME) (* AND MAKE THE FILE YOU COPIED THE WORKFILE! *)
347  10    7:0   488  END;
348  10    7:0   510
349  10    8:D     1  PROCEDURE READFILE;
350  10    8:0     0  BEGIN
351  10    8:1     0    CLEARSCREEN;           (* DUMB TERMINAL PATCH *)
352  10    8:1     3    WRITELN('>EDIT:');
353  10    8:1    25    WRITE('READING');
354  10    8:1    42    RESET(THEFILE);        (* WAS POTENTIALLY CLOSED BY BACKUP *)
355  10    8:1    48    IF BLOCKREAD(THEFILE,PAGEZERO,2)<>2 THEN ERROR('READING FILE',FATAL);
356  10    8:1    89    WRITE('.');
357  10    8:1    97    GETPAGES(RIGHTSTACK)
358  10    8:0    98  END;
359  10    8:0   114
360  10    8:0   114
361  10    8:0   114  (* PEOPLE WITH WORD MACHINES -- L O O K   A T   M E  ! ! *)
362  10    8:0   114
363  10    9:D     3  FUNCTION BYTESLEFT: INTEGER;
364  10    9:D     3  (* RETURNS THE NUMBER OF BYTES BETWEEN BLOCK AND LASTMP *)
365  10    9:0     0  BEGIN
366  10    9:1     0    BYTESLEFT:=(* DOUBLE FOR WORD MACHINES *) (ORD(SYSCOM^.LASTMP)-ORD(BLOCK))
367  10    9:0     8  END;
368  10    9:0    22
369  10    1:0     0  BEGIN
370  10    1:1     0    WITH PAGEZERO DO
```

```
371  10    1:2     0     BEGIN
372  10    1:2     0
373  10    1:2     0       (* INIT THE TRANSLATE TABLE *)
374  10    1:2     0
375  10    1:3     0       FILLCHAR(TRANSLATE,SIZEOF(TRANSLATE),ILLEGAL);
376  10    1:3    10
377  10    1:3    10       MAP('A',ADJUSTC);   MAP('B',BANISHC);   MAP('C',COPYC);
378  10    1:3    22       MAP('D',DELETEC);   MAP('F',FINDC);     MAP('I',INSERTC);
379  10    1:3    34       MAP('J',JUMPC);     MAP('L',LISTC);     MAP('M',MACRODEFC);
380  10    1:3    46       MAP('N',NEXTC);     MAP('P',PARAC);     MAP('Q',QUITC);
381  10    1:3    58       MAP('R',REPLACEC);  MAP('S',SETC);      MAP('V',VERIFYC);
382  10    1:3    70       MAP('X',XECUTEC);   MAP('Z',ZAPC);
383  10    1:3    78
384  10    1:3    78       MAP(',',REVERSEC);  MAP('>',FORWARDC);  MAP('.',FORWARDC);
385  10    1:3    90       MAP('+',FORWARDC);  MAP('-',REVERSEC);  MAP('?',DUMPC);
386  10    1:3   102       MAP('/',SLASHC);    MAP('=',EQUALC);    MAP('<',REVERSEC);
387  10    1:3   114
388  10    1:3   114
389  10    1:3   114       (* ARROWS *)
390  10    1:3   114
391  10    1:3   114       (* NEXTCOMMAND AND GETNUM HANDLE VT-52 STYLE VECTOR KEYS *)
392  10    1:3   114       IF SYSCOM^.CRTCTRL.ESCAPE=CHR(0) THEN
393  10    1:4   126         BEGIN
394  10    1:5   126           MAP(SYSCOM^.CRTINFO.LEFT,LEFT);
395  10    1:5   137           MAP(SYSCOM^.CRTINFO.DOWN,DOWN);
396  10    1:5   148           MAP(SYSCOM^.CRTINFO.RIGHT,RIGHT);
397  10    1:5   159           MAP(SYSCOM^.CRTINFO.UP,UP);
398  10    1:4   170         END;
399  10    1:3   170       MAP(SYSCOM^.CRTINFO.CHARDEL,LEFT);
400  10    1:3   181       MAP(CHR(EOL),ADVANCE); (* CR IS ADVANCE *)
401  10    1:3   185       MAP(CHR(HT),TAB);
402  10    1:3   189       MAP(CHR(SP),SPACE);
403  10    1:3   193
404  10    1:3   193
405  10    1:3   193       (* DIGITS *)
406  10    1:3   193
407  10    1:3   193       FOR CH:='0' TO '9' DO MAP(CH,DIGIT);
408  10    1:3   218
409  10    1:3   218
410  10    1:3   218       (* VARIABLE BUFFER SIZING... ADDED 17-JAN-78 *)
411  10    1:3   218
```

```
412  10    1:3   216    QUIT:=11000+        (* SIZEOF(EDITCORE)-SIZEOF(INITIALIZE) *)
413  10    1:3   221          512;         (* SLOP! *)
414  10    1:3   227    MARK(EBUF);
415  10    1:3   231    BLKS:=0;
416  10    1:3   234    REPEAT
417  10    1:4   234      NEW(BLOCK);
418  10    1:4   241      BLKS:=BLKS+1;
419  10    1:4   246      GAP:=BYTESLEFT-512        (* BYTESLEFT RETURNS THE # OF BYTES BETWEEN
420  10    1:4   250                                  THE POINTERS BLOCK AND LASTMP *)
421  10    1:3   250    UNTIL ((GAP>0) AND (GAP<QUIT)) OR (BLKS=63);
422  10    1:3   269    BUFSIZE:=BLKS*512-1;
423  10    1:3   278    IF BUFSIZE<0 THEN BUFSIZE:=32767;
424  10    1:3   288    NEW(ONEWD); ONEWD^:=0;  (* SENTINEL FOR END OF BUFFER - FOR M(UNCH *)
425  10    1:3   296
426  10    1:3   296
427  10    1:3   296    (* OPEN THE WORKFILE *)
428  10    1:3   296
429  10    1:3   296    LPAGE:=0; (* LEFT STACK EMPTY *)
430  10    1:3   300    RPAGE:=1; (* RIGHT STACK CONTAINS ALL OF THE WORKFILE *)
431  10    1:3   304    BUFCOUNT:=1;
432  10    1:3   307    CURSOR:=1;
433  10    1:3   310    CLEARSCREEN;
434  10    1:3   313    WRITELN('>EDIT:');
435  10    1:3   335    IF USERINFO.GOTSYM THEN
436  10    1:4   340      BEGIN
437  10    1:5   340        FILENAME:=CONCAT(USERINFO.SYMVID,':',USERINFO.SYMTID);
438  10    1:5   382        CLEANTITLE(FILENAME);
439  10    1:5   386        RESET(THEFILE,WFNAME);
440  10    1:5   397        IF IORESULT<>0 THEN ERROR('WORKFILE LOST.',FATAL)
441  10    1:4   421      END
442  10    1:3   424    ELSE
443  10    1:4   426      BEGIN
444  10    1:5   426        MSG:='NO WORKFILE IS PRESENT. FILE? ( <RET> FOR NO FILE ) ';
445  10    1:5   486        REPEAT
446  10    1:6   486          WRITE(MSG);
447  10    1:6   496          READLN(INPUT,FILENAME);
448  10    1:6   511          IF LENGTH(FILENAME)=0 THEN (* OPEN UP GOOD OL' SYSTEM.WRK.TEXT! *)
449  10    1:7   519            BEGIN
450  10    1:8   519              FILENAME:='*SYSTEM.WRK.TEXT';
451  10    1:8   542              CLEANTITLE(FILENAME);
452  10    1:8   546              FILLCHAR(EBUF^,BUFSIZE+1,CHR(0));
```

```
453  10    1:8   554                EBUF^[0]:=CHR(EOL);
454  10    1:8   558                FILLCHAR(PAGEZERO,SIZEOF(PAGEZERO),CHR(0));
455  10    1:8   568                REWRITE(THEFILE,WFNAME);
456  10    1:8   579                BACKFNAME:='';
457  10    1:3   587                IF IORESULT<>0 THEN ERROR('SYSTEM VOLUME NOT ON LINE',FATAL);
458  10    1:8   625                (* ESTABLISH THE LENGTH OF THE FILE AND LOCK THE FILE
459  10    1:8   625                   TO BE THE MAXIMUM EVEN LENGTH *)
460  10    1:8   625                FLENGTH:=FINDLENGTH(THEFILE);
461  10    1:8   635                IF ODD(FLENGTH) THEN FLENGTH:=FLENGTH-1;
462  10    1:8   648                IF BLOCKWRITE(THEFILE,BUFFER,1,2*FLENGTH-1)<>1
463  10    1:8   668                  THEN ERROR('FILE SYSTEM TERMINAL ERROR',FATAL);
464  10    1:8   705                CLOSE(THEFILE,LOCK);
465  10    1:8   712                WITH USERINFO DO
466  10    1:9   712                  BEGIN
467  10    1:0   712                    SYMVID:=SYVID; SYMTID:='SYSTEM.WRK.TEXT'; GOTSYM:=TRUE;
468  10    1:0   747                    OPENOLD(THEFILE,'*SYSTEM.WRK.CODE'); CLOSE(THEFILE,PURGE);
469  10    1:0   781                    GOTCODE:=FALSE; CODETID:=''
470  10    1:9   788                  END;
471  10    1:8   793                RESET(THEFILE,'*SYSTEM.WRK.TEXT');
472  10    1:8   820                RPAGE:=FLENGTH;
473  10    1:8   826                GOTO 1;
474  10    1:7   828              END;
475  10    1:6   828            CLEANTITLE(FILENAME);
476  10    1:6   832            OPENOLD(THEFILE,WFNAME);
477  10    1:6   843            MSG:='NOT PRESENT. FILE? ';
478  10    1:5   870          UNTIL IORESULT=0;
479  10    1:4   876        END;
480  10    1:4   876
481  10    1:4   876
482  10    1:4   876        (* FIND OUT THE LENGTH OF THE WORKFILE *)
483  10    1:4   876
484  10    1:3   876        FLENGTH:=FINDLENGTH(THEFILE);
485  10    1:3   886
486  10    1:3   886
487  10    1:3   886        (* IF DESIRED, COPY THE WORKFILE (MAXIMIZING EDITING ROOM) *)
488  10    1:3   886
489  10    1:3   886        BACKUP;
490  10    1:3   888
491  10    1:3   888
492  10    1:3   888        (* READ IN THE FILE *)
493  10    1:3   888
```

```
494  10    1:3   888        FILLCHAR(EBUF^,BUFSIZE+1,CHR(0));
495  10    1:3   896        EBUF^[0]:=CHR(EOL);
496  10    1:3   900        READFILE;
497  10    1:3   902        1: IF (EBUF^[BUFCOUNT-1]<>CHR(EOL)) OR (BUFCOUNT=1) THEN
498  10    1:4   915          BEGIN
499  10    1:5   915            EBUF^[BUFCOUNT]:=CHR(EOL);
500  10    1:5   919            BUFCOUNT:=BUFCOUNT+1;
501  10    1:4   924          END;
502  10    1:4   924
503  10    1:4   924
504  10    1:4   924        (* INITIALIZE EVERYTHING ELSE! *)
505  10    1:4   924
506  10    1:3   924        DIRECTION:='>';
507  10    1:3   927        LASTPAT:=1; (* INIT TO THE BEGINNING OF THE BUFFER (FOR EQUALC) *)
508  10    1:3   930        COPYOK:=FALSE;
509  10    1:3   934        LINE1PTR:=1;
510  10    1:3   938        WITH SYSCOM^.CRTINFO DO
511  10    1:4   946          BEGIN
512  10    1:5   946            ESC:=ORD(ALTMODE);
513  10    1:5   957            ETX:=ORD(EOF);
514  10    1:5   968            BS:=ORD(CHARDEL);
515  10    1:5   979            DEL:=ORD(LINEDEL);
516  10    1:5   990            SCREENWIDTH:=WIDTH-1;
517  10    1:5   998            SCREENHEIGHT:=HEIGHT-1;
518  10    1:5  1006            MIDDLE:=(SCREENHEIGHT DIV 2) + 1;
519  10    1:4  1014          END;
520  10    1:3  1014        SYSCOM^.MISCINFO.NOBREAK := TRUE;
521  10    1:3  1023        SDEFINED:=FALSE; TDEFINED:=FALSE; (* NO SUBSTRING OR TARGET *)
522  10    1:3  1031
523  10    1:3  1031
524  10    1:3  1031        (* SET UP PAGEZERO IF NEC. *)
525  10    1:3  1031
526  10    1:3  1031        DEFAULTPZ;
527  10    1:3  1033        REVISION:=REVISION+1;
528  10    1:3  1041
529  10    1:3  1041
530  10    1:2  1041      END(* WITH *);
531  10    1:2  1041
532  10    1:2  1041
533  10    1:2  1041    (* INITIALIZE THE KIND ARRAY FOR TOKEN FIND *)
534  10    1:2  1041
```

```
535  10    1:1  1041   FOR CH:=CHR(0) TO CHR(255) DO KIND[CH]:=ORD(CH);  (* MAKE THEM ALL UNIQUE *)
536  10    1:1  1071   FOR CH:='A' TO 'Z' DO KIND[CH]:=ORD('A');
537  10    1:1  1099   FOR CH:='A' TO 'Z' DO KIND[CH]:=ORD('A');
538  10    1:1  1127   FOR CH:='0' TO '9' DO KIND[CH]:=ORD('A');
539  10    1:1  1155   KIND[CHR(EOL)]:=ORD(' '); KIND[CHR(HT)] :=ORD(' ');
540  10    1:1  1169   FILLCHAR(BLANKAREA,SIZEOF(BLANKAREA),' '); (* FOR UNITWRITING BLANKS *)
541  10    1:1  1177   SAVETOP:=' ';  (* FOR BLANKCRT(1) - SAVES THE PROMPT OR MSG LINE *)
542  10    1:1  1183
543  10    1:0  1183 END(* INITIALIZE *);
544  10    1:0  1214
545  10    1:0  1214
546  10    1:0  1214 (*$TO U T*)
547  11    1:D     3 SEGMENT FUNCTION OUT: BOOLEAN;
548  11    1:D     3 LABEL 1,2;
549  11    1:D     3 TYPE
550  11    1:D     3   PHYLE=FILE;
551  11    1:D     3 VAR
552  11    1:D     3   SAVE: PTRTYPE;
553  11    1:D     4   RBNUM,LBNUM,MAXBLKSINBUF,BLKSREAD,I: INTEGER;
554  11    1:D     9   BUF: PACKED ARRAY [0..1023] OF CHAR;
555  11    1:D   521   FN: STRING;
556  11    1:D   562
557  11    2:D     1 PROCEDURE CHANGENAME(VAR F:PHYLE; T:STRING);
558  11    2:D    44 (* CHANGE THE TITLE OF F TO T.  NOTE: (1) THE FILE F MUST BE CLOSED WITH
559  11    2:D    44    CLOSE(F,LOCK), AND (2) THIS CODE RELIES ON A "SPECIAL FEATURE" IN THE
560  11    2:D    44    I/O SUBSYSTEM, NAMELY WHEN THE YEAR IS SET TO 100 THE TITLE GETS UPDATED
561  11    2:D    44    WHEN THE FILE IS CLOSED *)
562  11    2:D    44 VAR
563  11    2:D    44   COLON: INTEGER;
564  11    2:D    45   D: DATEREC;
565  11    2:D    46   FIBPA: PACKED ARRAY [0..57] OF CHAR;
566  11    2:0     0 BEGIN
567  11    2:0     0   (* MAKE SURE THAT THE FILENAME DOESN'T INCLUDE THE VOLUME NAME (OR "*") *)
568  11    2:1     0   COLON:=POS(':',T);
569  11    2:1    18   IF COLON>0 THEN DELETE(T,1,COLON);
570  11    2:1    32   IF T[1]='*' THEN DELETE(T,1,1);
571  11    2:1    47   MOVELEFT(F,FIBPA,58); (* TRANSFERS THE FIB FOR THE FILE F TO FIBPA *)
572  11    2:1    55   MOVELEFT(T,FIBPA[38],16);
573  11    2:1    64   WITH D DO BEGIN DAY:=2; MONTH:=3; YEAR:=100 END;
574  11    2:1    82   MOVELEFT(D,FIBPA[56],2);
575  11    2:1    91   MOVELEFT(FIBPA,F,58)
```

```
576  11    2:3     99 END;
577  11    2:3    112
578  11    3:0      1 PROCEDURE SETLASTBLOCK(LASTBLOCK:INTEGER);
579  11    3:0      2 (* KLUDGE CODE TO REMOVE BLOCKS FROM THE END OF THE WORKFILE *)
580  11    3:0      2 VAR FIBAREA:ARRAY [0..12] OF INTEGER;
581  11    3:0      0 BEGIN
582  11    3:1      0   MOVELEFT(THEFILE,FIBAREA,26);
583  11    3:1     10   FIBAREA[12]:=LASTBLOCK;
584  11    3:1     17   MOVELEFT(FIBAREA,THEFILE,26);
585  11    3:0     27 END;
586  11    3:0     40
587  11    1:0      0 BEGIN
588  11    1:1      0   OUT:=FALSE;
589  11    1:1      3   REPEAT
590  11    1:2      3     CLEARSCREEN;      (* DUMB TERMINAL PATCH *)
591  11    1:2      6     SAVETOP:='>QUIT:';
592  11    1:2     20     WRITELN(SAVETOP);
593  11    1:2     36     WRITELN('     U(PDATE THE WORKFILE AND LEAVE');
594  11    1:2     87     WRITELN('     E(XIT (BUT WORKFILE NOT UPDATED)');
595  11    1:2    140     WRITELN('     R(ETURN TO THE EDITOR WITHOUT DOING ANYTHING');
596  11    1:2    205     CH:=UCLC(GETCH);
597  11    1:1    217   UNTIL CH IN ['U','E','R'];
598  11    1:1    236   IF CH='R' THEN GOTO 2;
599  11    1:1    243   IF CH='E' THEN
600  11    1:2    248     BEGIN
601  11    1:3    248       OUT:=TRUE;
602  11    1:3    251       CLEARSCREEN;
603  11    1:3    254       CLOSE(THEFILE,PURGE);
604  11    1:3    261       IF LENGTH(BACKFNAME)>0 THEN
605  11    1:4    270         BEGIN
606  11    1:5    270           RESET(THEFILE,BACKFNAME);
607  11    1:5    281           IF IORESULT=0 THEN
608  11    1:6    297             BEGIN
609  11    1:7    287               CHANGENAME(THEFILE,WFNAME);
610  11    1:7    295               CLOSE(THEFILE,LOCK);
611  11    1:6    302             END
612  11    1:5    302           ELSE
613  11    1:6    304             WRITELN('BACKUP FILE NOT PRESENT (TRIED TO REMOVE IT).');
614  11    1:5    365           GOTO 2
615  11    1:4    367         END
616  11    1:3    367       ELSE GOTO 2;
```

```
617  11    1:2   371     END;
618  11    1:1   371   BLANKCRT(1);
619  11    1:1   375   CURSOR:=BUFCOUNT+199;  (* TAKES CARE OF THE SLOP! *)
620  11    1:1   382   WRITE('WRITING');
621  11    1:1   399   PUTPAGES(LEFTSTACK);
622  11    1:1   403   PAGEZERO.LASTUPD:=THEDATE; (* RESET LAST UPDATE DATE *)
623  11    1:1   411   IF LPAGE+1=RPAGE THEN BEGIN OUT:=TRUE; CLEARSCREEN; GOTO 2 END;
624  11    1:1   430   IF LPAGE+1>RPAGE THEN ERROR('LPAGE+1>RPAGE',FATAL);
625  11    1:1   461   LBNUM:=2*(LPAGE+1);
626  11    1:1   470   RBNUM:=2*RPAGE;
627  11    1:1   477   MAXBLKSINBUF:=BUFSIZE DIV 512;
628  11    1:1   484   REPEAT
629  11    1:2   484     WRITE('*');
630  11    1:2   492     BLKSREAD:=BLOCKREAD(THEFILE,EBUF^,MAXBLKSINBUF,RBNUM);
631  11    1:2   507     IF IORESULT<>0 THEN GOTO 1;
632  11    1:2   515     IF BLKSREAD<>0 THEN
633  11    1:3   520       BEGIN
634  11    1:4   520         IF BLOCKWRITE(THEFILE,EBUF^,BLKSREAD,LBNUM)<>BLKSREAD THEN GOTO 1;
635  11    1:4   539         IF IORESULT<>0 THEN GOTO 1
636  11    1:3   547       END;
637  11    1:2   547     LBNUM:=LBNUM+BLKSREAD;
638  11    1:2   552     RBNUM:=RBNUM+BLKSREAD
639  11    1:1   553   UNTIL BLKSREAD=0;
640  11    1:1   562   SETLASTBLOCK(2*(LPAGE+1+FLENGTH-RPAGE));
641  11    1:1   579   (* COMPENSATE FOR GAP FILLED IN *)
642  11    1:1   579   WITH PAGEZERO DO
643  11    1:2   579     BEGIN
644  11    1:3   579       FOR I:=0 TO COUNT-1 DO
645  11    1:4   597         IF PAGENO[I]>=RPAGE THEN PAGENO[I]:=PAGENO[I]-(RPAGE-LPAGE)+1;
646  11    1:2   641     END;
647  11    1:1   641   IF BLOCKWRITE(THEFILE,PAGEZERO,2,0)<>2 THEN GOTO 1;
648  11    1:1   662   OUT:=TRUE;
649  11    1:1   665   WRITELN;
650  11    1:1   671   WRITELN('THE WORKFILE, ',WFNAME,
651  11    1:1   705           ', IS ',2*(LPAGE+1+FLENGTH-RPAGE),' BLOCKS LONG.');
652  11    1:1   771   IF LENGTH(BACKFNAME)>0 THEN WRITE('THE BACKUP FILE IS ',BACKFNAME);
653  11    1:1   819   CLOSE(THEFILE,LOCK);
654  11    1:1   826   GOTO 2;
655  11    1:1   828 1:ERROR('WRITING OUT THE FILE',FATAL);
656  11    1:1   855 2:END;
657  11    1:1   878
```

```
658  11    1:1   878
659  11    1:1   878
660  11    1:1   873 (*$TC O P Y F I L E*)
661  12    1:0     1 SEGMENT PROCEDURE COPYFILE;
662  12    1:0     1 VAR
663  12    1:0     1   STARTPAGE,STOPPAGE,STARTOFFSET,STOPOFFSET,
664  12    1:0     1   LEFTPART,PAGE,NOTNULLS,THEREST,LMOVE: INTEGER;
665  12    1:0    10   DONE,OVFLW: BOOLEAN;
666  12    1:0    12   BUFR: PACKED ARRAY [0..1023] OF CHAR;
667  12    1:0   524   STARTMARK,STOPMARK: PACKED ARRAY [0..7] OF CHAR;
668  12    1:0   532   FN: STRING;
669  12    1:0   573   F: FILE;
670  12    1:0   613
671  12    2:0     1 PROCEDURE ERRMARKER;
672  12    2:0     0 BEGIN
673  12    2:1     0   ERROR('IMPROPER MARKER SPECIFICATION.',NONFATAL);
674  12    2:1    37   EXIT(COPYFILE)
675  12    2:0    41 END;
676  12    2:0    54
677  12    3:0     1 PROCEDURE UNSPLITBUF;
678  12    3:0     1 (* STICH THE BUFFER BACK TOGETHER AGAIN. *)
679  12    3:0     1 VAR BOGOSITY: PTRTYPE;
680  12    3:0     0 BEGIN
681  12    3:1     0   MOVELEFT(EBUF^[THEREST],EBUF^[CURSOR],LMOVE);
682  12    3:1    11   READJUST(LEFTPART+1,CURSOR-(LEFTPART+1));
683  12    3:1    26   BUFCOUNT:=BUFCOUNT+CURSOR-(LEFTPART+1);
684  12    3:1    37   (* CHECK THAT TWO DLE'S IN A ROW HAVEN'T BEEN GENERATED *)
685  12    3:1    37   CHECKINDENT(CURSOR);
686  12    3:1    42   BOGOSITY:=LEFTPART+1;
687  12    3:1    49   CHECKINDENT(BOGOSITY);
688  12    3:0    54 END;
689  12    3:0    66
690  12    4:0     1 PROCEDURE READERR;
691  12    4:0     0 BEGIN
692  12    4:1     0   ERROR('MARKER EXCEEDS FILE BOUNDS.',NONFATAL);
693  12    4:1    34   UNSPLITBUFF;
694  12    4:1    36   CENTERCURSOR(TRASH,MIDDLE,TRUE);
695  12    4:1    46   EXIT(COPYFILE)
696  12    4:0    50 END;
697  12    4:0    62
698  12    5:0     1 PROCEDURE SPLITBUF;
```

```
699  12    5:D     1 (* SPLIT THE BUFFER AT THE CURSOR.  THEREST POINTS TO THE RIGHT PART, LMOVE
700  12    5:D     1    IS THE LENGTH OF THE RIGHT PART, LEFTPART POINTS TO THE END OF THE 'LEFT
701  12    5:D     1    PART', AND CURSOR REMAINS UNCHANGED. *)
702  12    5:0     0 BEGIN
703  12    5:1     0   THEREST:=BUFSIZE-(BUFCOUNT-CURSOR);
704  12    5:1     8   LMOVE:=BUFCOUNT-CURSOR+1;
705  12    5:1    16   LEFTPART:=CURSOR-1;
706  12    5:1    22   MOVERIGHT(EBUF^[CURSOR],EBUF^[THEREST],LMOVE)
707  12    5:0    33 END;
708  12    5:0    46
709  12    6:D     1 PROCEDURE PARSEFN;
710  12    6:D     1 VAR I,LPTR,RPTR,COMMA: INTEGER;
711  12    6:D     5     MARK: STRING;
712  12    6:0     0 BEGIN
713  12    6:1     0   LPTR:=POS('[',FN);
714  12    6:1    15   IF LPTR=0 THEN
715  12    6:2    20     BEGIN (* WHOLE FILE *)
716  12    6:3    20       STARTMARK:='         ';
717  12    6:3    37       STOPMARK:= '         '
718  12    6:2    41     END
719  12    6:1    54   ELSE
720  12    6:2    56     BEGIN
721  12    6:3    56       RPTR:=POS(']',FN);
722  12    6:3    71       IF (RPTR=0) OR (RPTR<LPTR) OR (RPTR<>LENGTH(FN)) THEN ERRMARKER;
723  12    6:3    91       MARK:=COPY(FN,LPTR+1,RPTR-LPTR-1); (* STUFF BETWEEN THE BRACKETS *)
724  12    6:3   114       FN:=COPY(FN,1,LPTR-1);
725  12    6:3   135       COMMA:=POS(',',MARK);
726  12    6:3   148       IF COMMA=0 THEN ERRMARKER;
727  12    6:3   155       I:=LENGTH(MARK)-COMMA; (* SECOND MARKER PTR *)
728  12    6:3   163       MOVELEFT(MARK[1],STARTMARK,MIN(8,COMMA-1));
729  12    6:3   182       FILLCHAR(STARTMARK[COMMA-1],MAX(0,8-(COMMA-1)),' ');
730  12    6:3   203       MOVELEFT(MARK[COMMA+1],STOPMARK,MIN(I,8));
731  12    6:3   222       FILLCHAR(STOPMARK[I],MAX(0,8-I),' ')
732  12    6:2   239     END;
733  12    6:1   239   FOR I:=0 TO 7 DO STARTMARK[I]:=UCLC(STARTMARK[I]);
734  12    6:1   275   FOR I:=0 TO 7 DO STOPMARK [I]:=UCLC(STOPMARK[I]);
735  12    6:1   311   FOR I:=1 TO LENGTH(FN) DO FN[I]:=UCLC(FN[I]);
736  12    6:1   352   IF ((POS('.TEXT',FN)<>LENGTH(FN)-4) OR
737  12    6:1   378      (LENGTH(FN)<=4)) AND (FN[LENGTH(FN)]<>',') THEN
738  12    6:2   403     FN:=CONCAT(FN,'.TEXT');
739  12    6:1   438   IF FN[LENGTH(FN)]=',' THEN D  ETE(FN,LENGTH(FN),1);
```

```
740  12    6:0   467 END;
741  12    6:0   488
742  12    7:0     1 PROCEDURE STUFFIT(START,STOP:INTEGER);
743  12    7:0     3 (* PUT THE CONTENTS OF BUFR INTO EBUF.  OVFLW IS SET TO TRUE WHEN THERE IS
744  12    7:0     3    NO MORE ROOM IN THE BUFFER. *)
745  12    7:0     3 VAR AMOUNT: INTEGER;
746  12    7:0     0 BEGIN
747  12    7:1     0   IF START<=STOP THEN
748  12    7:2     5     BEGIN
749  12    7:3     5       AMOUNT:=STOP-START+1;
750  12    7:3    12       IF CURSOR+AMOUNT+250(*SLOP*)>=THEREST THEN
751  12    7:4    25         BEGIN
752  12    7:5    25           ERROR('BUFFER OVERFLOW.',NONFATAL);
753  12    7:5    48           CURSOR:=LEFTPART+1;
754  12    7:5    55           UNSPLITBUFF;
755  12    7:5    57           EXIT(COPYFILE)
756  12    7:4    61         END
757  12    7:3    61       ELSE
758  12    7:4    63         BEGIN
759  12    7:5    63           MOVELEFT(BUFR[START],EBUF^[CURSOR],AMOUNT);
760  12    7:5    72           CURSOR:=CURSOR+AMOUNT
761  12    7:4    73         END
762  12    7:2    77     END
763  12    7:0    77 END;
764  12    7:0    90
765  12    8:0     1 PROCEDURE GETNEXT;
766  12    8:0     0 BEGIN
767  12    8:1     0   DONE:=BLOCKREAD(F,BUFR,2,PAGE+PAGE)<>2;
768  12    8:1    27   IF IORESULT<>0 THEN
769  12    8:2    33     BEGIN
770  12    8:3    33       ERROR('BAD DISK TRANSFER',NONFATAL);
771  12    8:3    57       CURSOR:=LEFTPART+1;
772  12    8:3    64       UNSPLITBUF;
773  12    8:3    66       EXIT(COPYFILE)
774  12    8:2    70     END;
775  12    8:1    70   WRITE('.');
776  12    8:1    78   IF NOT DONE THEN NOTNULLS:=SCAN(-1024,<>CHR(0),BUFR[1023])+1024
777  12    8:1    99   ELSE NOTNULLS:=0;
778  12    8:1   112   PAGE:=PAGE+1;
779  12    8:0   120 END;
780  12    8:0   132
```

```
781  12    9:0     1 PROCEDURE CHKOVFLW;
782  12    9:0     0 BEGIN
783  12    9:1     0   IF (STOPOFFSET>=NOTNULLS) AND (STOPPAGE<PAGE) THEN
784  12    9:2    17     BEGIN
785  12    9:3    17       STOPPAGE:=STOPPAGE+1;
786  12    9:3    25       STOPOFFSET:=STOPOFFSET-NOTNULLS;
787  12    9:2    35     END;
788  12    9:0    35 END;
789  12    9:0    48
790  12   10:0     1 PROCEDURE FINDMARKERS;
791  12   10:0     1 (* GIVEN STARTMARK AND STOPMARK FIND OUT THEIR PAGE NUMBERS AND OFFSETS *)
792  12   10:0     1 VAR
793  12   10:0     1   PZ: HEADER;
794  12   10:0   513
795  12   11:0     1   PROCEDURE SEARCH(MNAME:NAME;VAR OFF,PNUM: INTEGER);
796  12   11:0     8   VAR
797  12   11:0     8     I: INTEGER;
798  12   11:0     0   BEGIN
799  12   11:1     0     I:=0;
800  12   11:1     8     WHILE (I<PZ.COUNT) AND (MNAME<>PZ.NAME[I]) DO I:=I+1;
801  12   11:1    34     IF MNAME<>PZ.NAME[I] THEN
802  12   11:2    47       BEGIN
803  12   11:3    47         ERROR('MARKER NOT THERE.',NONFATAL);
804  12   11:3    71         UNSPLITBUFF;
805  12   11:3    73         EXIT(COPYFILE)
806  12   11:2    77       END;
807  12   11:1    77     OFF:=PZ.POFFSET[I];
808  12   11:1    86     PNUM:=PZ.PAGEN[I];
809  12   11:1    95     IF PNUM=0 THEN
810  12   11:2   101       BEGIN OFF:=OFF-1; PNUM:=1 END; (* KLUDGE TO MAINTAIN COMPATIBILITY *)
811  12   11:0   110   END;
812  12   11:0   124
813  12   10:0     0 BEGIN(* FINDMARKERS *)
814  12   10:1     0   STARTPAGE:=1;    STARTOFFSET:=0;   (* DEFAULT VALUES *)
815  12   10:1     8   STOPPAGE:=32767; STOPOFFSET:=32767;
816  12   10:1    20   IF (STARTMARK<>'          ') OR (STOPMARK<>'          ') THEN
817  12   10:2    59     BEGIN
818  12   10:3    59       IF BLOCKREAD(F,PZ,2,0)<>2 THEN READERR;
819  12   10:3    80       IF STARTMARK<>'          ' THEN SEARCH(STARTMARK,STARTOFFSET,STARTPAGE);
820  12   10:3   112       IF STOPMARK<>'          ' THEN SEARCH(STOPMARK,STOPOFFSET,STOPPAGE)
821  12   10:2   142     END
```

```
822  12    10:0    144 END;
823  12    10:0    156
824  12     1:0      0 BEGIN
825  12     1:1      0   PROMPTLINE:=' COPY: FROM WHAT FILE[MARKER,MARKER]? ';
826  12     1:1     59   REPEAT
827  12     1:2     59     PROMPT;
828  12     1:2     62     READLN(FN);
829  12     1:2     78     IF LENGTH(FN)=0 THEN EXIT(COPYFILE);
830  12     1:2     91     PARSEFN;
831  12     1:2     93     RESET(F,FN);
832  12     1:2    104     PROMPTLINE:=' COPY: FILE NOT PRESENT. FILENAME? ';
833  12     1:1    147   UNTIL IORESULT=0;
834  12     1:1    153   PROMPTLINE:=' COPY'; PROMPT;
835  12     1:1    169   SPLITBUF;
836  12     1:1    171   FINDMARKERS;
837  12     1:1    173   PAGE:=STARTPAGE;
838  12     1:1    176   GETNEXT;
839  12     1:1    178   WHILE (STARTOFFSET>=NOTNULLS) AND NOT DONE DO
840  12     1:2    186     BEGIN
841  12     1:3    186       CHKOVFLW;
842  12     1:3    188       STARTOFFSET:=STARTOFFSET-NOTNULLS;
843  12     1:3    193       GETNEXT;
844  12     1:2    195     END;
845  12     1:1    197   IF (STOPPAGE<PAGE) AND (STOPOFFSET<NOTNULLS) THEN
846  12     1:2    206     STUFFIT(STARTOFFSET,MIN(NOTNULLS-1,STOPOFFSET-1))
847  12     1:1    218   ELSE
848  12     1:2    222     STUFFIT(STARTOFFSET,NOTNULLS-1);
849  12     1:1    228   WHILE ((STOPPAGE>=PAGE) OR (STOPOFFSET>=NOTNULLS)) AND NOT DONE DO
850  12     1:2    240     BEGIN
851  12     1:3    240       CHKOVFLW;
852  12     1:3    242       GETNEXT;
853  12     1:3    244       IF (STOPPAGE<PAGE) AND (STOPOFFSET<NOTNULLS) THEN
854  12     1:4    253         STUFFIT(0,MIN(NOTNULLS-1,STOPOFFSET-1))
855  12     1:3    265       ELSE
856  12     1:4    269         STUFFIT(0,NOTNULLS-1)
857  12     1:2    273     END;
858  12     1:1    277   IF IORESULT<>0 THEN ERROR('DISK ERROR.',NONFATAL);
859  12     1:1    301   UNSPLITBUF;
860  12     1:1    303   CENTERCURSOR(TRASH,MIDDLE,TRUE);
861  12     1:1    313   CLOSE(F);
862  12     1:0    320 END;
```

```
863  12    1:0   346
864  12    1:0   346 (*$TE N v I R O N M E N T*)
865  13    1:0     1 SEGMENT PROCEDURE ENVIRONMENT;
866  13    1:0     1 VAR
867  13    1:0     1   I: INTEGER;
868  13    1:0     2
869  13    2:0     1   PROCEDURE WRITEDATE(THEDATE:DATEREC);
870  13    2:0     3   (* WRITE OUT (IN TEXT) THE DATE.  PLEASE NOTE THE RESTRAINT INVOLVED IN
871  13    2:0     3      NOT PUTTING IN MY BIRTHDAY! (RSK) *)
872  13    2:0     3   VAR T: STRING;
873  13    2:0     0   BEGIN
874  13    2:1     0     WITH THEDATE DO
875  13    2:2     5       BEGIN
876  13    2:3     5         IF MONTH=0 THEN WRITE('NONE')
877  13    2:3    28         ELSE
878  13    2:4    30           BEGIN
879  13    2:5    30             IF (MONTH=12) AND (DAY=25) THEN
880  13    2:6    47               WRITE('CHRISTMAS')
881  13    2:5    66             ELSE
882  13    2:6    68               IF (MONTH=1) AND (DAY=1) THEN
883  13    2:7    85                 WRITE('NEW YEARS')
884  13    2:6   104               ELSE
885  13    2:7   106                 IF (MONTH=10) AND (DAY=31) THEN
886  13    2:8   123                   WRITE('HALLOWEEN')
887  13    2:7   142                 ELSE
888  13    2:8   144                   BEGIN
889  13    2:9   144                     CASE MONTH OF
890  13    2:9   151                       1: T:='JANUARY';
891  13    2:9   167                       2: T:='FEBRUARY';
892  13    2:9   184                       3: T:='MARCH';
893  13    2:9   198                       4: T:='APRIL';
894  13    2:9   212                       5: T:='MAY';
895  13    2:9   224                       6: T:='JUNE';
896  13    2:9   237                       7: T:='JULY';
897  13    2:9   250                       8: T:='AUGUST';
898  13    2:9   265                       9: T:='SEPTEMBER';
899  13    2:9   283                       10:T:='OCTOBER';
900  13    2:9   299                       11:T:='NOVEMBER';
901  13    2:9   316                       12:T:='DECEMBER'
902  13    2:9   318                     END;
903  13    2:9   364                     WRITE(T,'  DAY);
```

```
904  13    2:3   393                    END;
905  13    2:5   393                WRITE(', ',YEAR+1900);
906  13    2:4   421              END;
907  13    2:2   421          END;
908  13    2:0   421      END;
909  13    2:0   446
910  13    3:0     1      PROCEDURE ERASE10;
911  13    3:0     1      VAR I: INTEGER;
912  13    3:0     0      BEGIN
913  13    3:1     0        WRITE(' ':10);
914  13    3:1     8        FOR I:=1 TO 10 DO WRITE(CHR(BSPCE));
915  13    3:0    34      END;
916  13    3:0    48
917  13    4:0     1      PROCEDURE BOOL(B:BOOLEAN);
918  13    4:0     0      BEGIN
919  13    4:1     0        IF B THEN WRITE('TRUE') ELSE WRITE('FALSE');
920  13    4:1    34        WRITELN
921  13    4:0    34      END;
922  13    4:0    52
923  13    5:0     3      FUNCTION GETBOOL: BOOLEAN;
924  13    5:0     3      VAR CH: CHAR;
925  13    5:0     0      BEGIN
926  13    5:1     0        ERASE10; CH:=UCLC(GETCH);
927  13    5:1    14        WHILE NOT (CH IN ['T','F']) DO
928  13    5:2    35          BEGIN
929  13    5:3    35            WRITE('T OR F');
930  13    5:3    51            FOR TRASH:=0 TO 5 DO WRITE(CHR(BS));
931  13    5:3    85            CH:=UCLC(GETCH)
932  13    5:2    90          END;
933  13    5:1    99        IF CH='T' THEN
934  13    5:2   104          BEGIN
935  13    5:3   104            WRITE('TRUE  ');
936  13    5:3   120            GETBOOL:=TRUE
937  13    5:2   120          END
938  13    5:1   123        ELSE
939  13    5:2   125          BEGIN
940  13    5:3   125            WRITE('FALSE ');
941  13    5:3   141            GETBOOL:=FALSE
942  13    5:2   141          END;
943  13    5:0   144    END;
944  13    5:0   160
```

```
945  13    6:0     3   FUNCTION GETINT: INTEGER;
946  13    6:0     3   VAR
947  13    6:0     3     CH:CHAR;
948  13    6:0     4     N: INTEGER;
949  13    6:0     0   BEGIN
950  13    6:1     0     ERASE10;
951  13    6:1     2     N:=0;
952  13    6:1     5     REPEAT
953  13    6:2     5       REPEAT
954  13    6:3     5         CH:=GETCH;
955  13    6:3    12         IF NOT (CH IN ['0'..'9',CHR(SP),CHR(CR)])
956  13    6:3    31           THEN WRITE('#',CHR(BELL),CHR(BS));
957  13    6:2    60       UNTIL CH IN ['0'..'9',CHR(SP),CHR(CR)];
958  13    6:2    81       IF CH IN ['0'..'9'] THEN
959  13    6:3    96         BEGIN
960  13    6:4    96           WRITE(CH);
961  13    6:4   104           IF N<1000 THEN N:=N*10+ORD(CH)-ORD('0')
962  13    6:3   117         END;
963  13    6:1   120     UNTIL CH IN [CHR(SP),CHR(CR)];
964  13    6:1   129     GETINT:=N; WRITE('  ')
965  13    6:0   144   END;
966  13    6:0   160
967  13    7:0     1 PROCEDURE TABSET;
968  13    7:0     1 VAR
969  13    7:0     1   X,I,NUMTIMES: INTEGER;
970  13    7:0     4
971  13    8:0     1   PROCEDURE SETIT(CH:CHAR);
972  13    8:0     2   (* SET THE TABSTOP ACCORDING TO THE CHARACTER PASSED *)
973  13    8:0     0   BEGIN
974  13    8:1     0     WITH PAGEZERO DO
975  13    8:2     0       CASE CH OF
976  13    8:2     3         'N','-': BEGIN CH:='-'; TABSTOP[X]:=NONE; END;
977  13    8:2    19         'L':     TABSTOP[X]:=LEFTJUST;
978  13    8:2    32         'R':     TABSTOP[X]:=RIGHTJUST;
979  13    8:2    45         'D':     TABSTOP[X]:=DECIMALSTOP
980  13    8:2    54       END;
981  13    8:1   142     WRITE(CH);
982  13    8:0   150   END;
983  13    8:0   162
984  13    7:0     0 BEGIN
985  13    7:1     0   WITH PAGEZERO DO
```

```
 986  13    7:2     2      BEGIN
 987  13    7:3     0        CLEARSCREEN;
 988  13    7:3     3        WRITELN(
 989  13    7:3     3 'SET TABS: <RIGHT,LEFT VECTORS>  C(OL#  E(N(O R(IGHT L(EFT D(ECIMAL STOP]  <ETX>'
 990  13    7:3     6        );
 991  13    7:3    97        WRITELN;
 992  13    7:3   103        FOR I:=0 TO SCREENWIDTH DO
 993  13    7:4   114          CASE TABSTOP[I] OF
 994  13    7:4   124            NONE: WRITE('-');
 995  13    7:4   134            LEFTJUST: WRITE('L');
 996  13    7:4   144            RIGHTJUST: WRITE('R');
 997  13    7:4   154            DECIMALSTOP: WRITE('D')
 998  13    7:4   162          END;
 999  13    7:3   187        X:=0;
1000  13    7:3   190        GOTOXY(4,4); WRITE('COLUMN #');
1001  13    7:3   213        REPEAT
1002  13    7:4   213          GOTOXY(12,4); WRITE(X+1:3);
1003  13    7:4   228          GOTOXY(X,2);
1004  13    7:4   233          CH:=UCLC(GETCH);
1005  13    7:4   245          NUMTIMES:=GETNUM; (* ALSO SETS COMMAND *)
1006  13    7:4   252          IF CH IN E'N','D','L','R','-'] THEN SETIT(CH)
1007  13    7:4   273          ELSE
1008  13    7:5   277            IF CH='C' THEN
1009  13    7:6   282              BEGIN
1010  13    7:7   282                GOTOXY(12,4);
1011  13    7:7   287                X:=MAX(0,MIN(GETINT,SCREENWIDTH+1)-1);
1012  13    7:6   309              END
1013  13    7:5   309            ELSE
1014  13    7:6   311              IF COMMAND=LEFT THEN X:=MAX(0,X-NUMTIMES)
1015  13    7:6   320              ELSE
1016  13    7:7   329                IF COMMAND=RIGHT THEN X:=MIN(X+NUMTIMES,SCREENWIDTH)
1017  13    7:7   338                ELSE
1018  13    7:8   347                  IF NOT (CH IN E(CHR(ETX),' ']) THEN WRITE(CHR(BELL));
1019  13    7:3   374        UNTIL CH=CHR(ETX);
1020  13    7:3   381        CH:='$'; (* SO WE DON'T FALL OUT ALL OF THE WAY! *)
1021  13    7:2   384      END;
1022  13    7:0   384 END;
1023  13    7:0   400
1024  13    9:0     1 PROCEDURE WRITEMENU;
1025  13    9:0     0 BEGIN
1026  13    9:1     0   WITH PAGEZERO DO
```

```
1027  13     9:2      0     BEGIN
1028  13     9:3      0       WRITELN;
1029  13     9:3      6       WRITE(        '    A(UTO INDENT  '); BOOL(AUTOINDENT);
1030  13     9:3     39       WRITE(        '    F(ILLING      '); BOOL(FILLING);
1031  13     9:3     72       WRITE(        '    L(EFT MARGIN  '); WRITELN(LMARGIN+1);
1032  13     9:3    118       WRITE(        '    R(IGHT MARGIN '); WRITELN(RMARGIN+1);
1033  13     9:3    164       WRITE(        '    P(ARA MARGIN  '); WRITELN(PARAMARGIN+1);
1034  13     9:3    210       WRITE(        '    C(OMMAND CH   '); WRITELN(RUNOFFCH);
1035  13     9:3    254       WRITE(        '    S(ET TABSTOPS '); WRITELN;
1036  13     9:3    288       WRITE(        '    T(OKEN DEF    '); BOOL(TOKDEF);
1037  13     9:3    321       WRITELN;
1038  13     9:3    327       WRITELN(BUFCOUNT,' BYTES USED, ',BUFSIZE-BUFCOUNT+1,' AVAILABLE.');
1039  13     9:3    397       WRITELN('THERE ARE ',LPAGE,' PAGES IN THE LEFT STACK, AND ',
1040  13     9:3    467               FLENGTH-RPAGE,' PAGES IN THE RIGHT STACK.');
1041  13     9:3    523       WRITELN('YOU HAVE ',RPAGE-LPAGE-1,' PAGES OF ROOM,',
1042  13     9:3    583            ' AND AT MOST ',(BUFCOUNT DIV 960)+1,' PAGES WORTH IN THE BUFFER.');
1043  13     9:3    663       WRITELN;
1044  13     9:2    669     END;
1045  13     9:0    669 END;
1046  13     9:0    682
1047  13    10:D      1 PROCEDURE WRITEINFO;
1048  13    10:0      0 BEGIN
1049  13    10:1      0   WITH PAGEZERO DO
1050  13    10:2      0     BEGIN
1051  13    10:3      0       IF SDEFINED OR TDEFINED THEN
1052  13    10:4      9         BEGIN
1053  13    10:5      9           WRITELN('    PATTERNS:');
1054  13    10:5     38           IF TDEFINED THEN WRITE('      <TARGET>= ''',TARGET:TLENGTH,'''');
1055  13    10:5     93           IF SDEFINED THEN WRITE(', <SUBST>=  ''',SUBSTRING:SLENGTH,'''');
1056  13    10:5    144           WRITELN; WRITELN;
1057  13    10:4    156         END;
1058  13    10:3    156       IF COUNT>0 THEN WRITELN('    MARKERS:');
1059  13    10:3    191       WRITE('  ');
1060  13    10:3    203       FOR I:=0 TO COUNT-1 DO
1061  13    10:4    221         BEGIN
1062  13    10:5    221           WRITE('   ');
1063  13    10:5    234           IF PAGEN[I]=-1 THEN
1064  13    10:6    248             WRITE(' ')
1065  13    10:5    256           ELSE
1066  13    10:6    258             IF PAGEN[I]<=LPAGE THEN WRITE('<') ELSE WRITE('>');
1067  13    10:5    291           WRITE(NAME[I]);
```

```
1068  13    10:5    307            IF (I<>COUNT-1) AND ((I+1) MOD 5=0) THEN
1069  13    10:6    328              BEGIN WRITELN; WRITE('  ') END
1070  13    10:4    346          END;
1071  13    10:3    356        WRITELN;
1072  13    10:3    362        WRITELN;
1073  13    10:3    368        WRITE('    CREATED '); WRITEDATE(CREATED);
1074  13    10:3    395        WRITE('; LAST UPDATED '); WRITEDATE(LASTUPD);
1075  13    10:3    425        WRITE(' (REVISION ',REVISION,').');
1076  13    10:2    468      END;
1077  13    10:0    488  END;
1078  13    10:0    486
1079  13     1:0      0  BEGIN
1080  13     1:1      0    WITH PAGEZERO DO
1081  13     1:2      0      BEGIN
1082  13     1:3      0        CLEARSCREEN;
1083  13     1:3      3        PROMPTLINE:= ' ENVIRONMENT: [OPTIONS] <SPACEBAR> TO LEAVE';
1084  13     1:3     54        PROMPT; NEEDPROMPT:=TRUE;
1085  13     1:3     61        WRITEMENU;
1086  13     1:3     63        WRITEINFO;
1087  13     1:3     65        GOTOXY(LENGTH(PROMPTLINE),0);
1088  13     1:3     74        REPEAT
1089  13     1:4     74          CH:=UCLC(GETCH);
1090  13     1:4     86          IF NOT (CH IN ['A','C','F','L','P','R','S','T',' ',CHR(CR)])
1091  13     1:4    107          THEN
1092  13     1:5    110            BEGIN ERROR('NOT OPTION',NONFATAL); PROMPT; END
1093  13     1:4    130          ELSE
1094  13     1:5    132            CASE CH OF
1095  13     1:5    135              'A': BEGIN GOTOXY(18,1); AUTOINDENT:=GETBOOL END;
1096  13     1:5    149              'F': BEGIN GOTOXY(18,2); FILLING:=GETBOOL END;
1097  13     1:5    163              'L': BEGIN GOTOXY(18,3); LMARGIN:=MAX(0,GETINT-1) END;
1098  13     1:5    185              'R': BEGIN GOTOXY(18,4); RMARGIN:=MAX(0,GETINT-1) END;
1099  13     1:5    207              'P': BEGIN GOTOXY(18,5); PARAMARGIN:=MAX(0,GETINT-1) END;
1100  13     1:5    229              'C': BEGIN GOTOXY(18,6); READ(RUNOFFCH) END;
1101  13     1:5    245              'S': BEGIN
1102  13     1:7    245                     TABSET; (* NEW SCREEN DISPLAYED *)
1103  13     1:7    247                     CLEARSCREEN;
1104  13     1:7    250                     PROMPT;
1105  13     1:7    253                     WRITEMENU;
1106  13     1:7    255                     WRITEINFO;
1107  13     1:7    257                     GOTOXY(LENGTH(PROMPTLINE),0)
1108  13     1:6    266                   END;
```

```
1109  13     1:5    268            'T': BEGIN GOTOXY(18,8); TOKDEF:=GETBOOL END
1110  13     1:5    280          END;
1111  13     1:4    330        GOTOXY(LENGTH(PROMPTLINE),0);
1112  13     1:3    339      UNTIL CH IN [' ',CHR(CR)];
1113  13     1:3    355      REDISPLAY;
1114  13     1:2    358    END;
1115  13     1:0    358 END;
1116  13     1:0    378
1117  13     1:0    378
1118  13     1:0    378
1119  13     1:0    378 (*$TP U T S Y N T A X*)
1120  14     1:D      1 SEGMENT PROCEDURE PUTSYNTAX;
1121  14     1:D      1 VAR
1122  14     1:D      1   D0,D1,D2,BLK,PTR,COLON: INTEGER;
1123  14     1:D      7   T,C:PACKED ARRAY [0..2] OF CHAR;
1124  14     1:D     11   BUF:PACKED ARRAY [0..1023] OF CHAR;
1125  14     1:D    523   F: FILE;
1126  14     1:D    563
1127  14     2:D      1 PROCEDURE PUTNUM;
1128  14     2:0      0 BEGIN
1129  14     2:1      0   MSG:='SYNTAX ERROR #'; PUTMSG;
1130  14     2:1     25   WRITE(USERINFO.ERRNUM,'. TYPE <SP>');
1131  14     2:0     56 END;
1132  14     2:0     68
1133  14     1:0      0 BEGIN (* PUTSYNTAX *)
1134  14     1:1      0   WITH USERINFO DO
1135  14     1:2     13     BEGIN
1136  14     1:3     13       OPENOLD(F,'*SYSTEM.SYNTAX');
1137  14     1:3     38       IF IORESULT<>0 THEN PUTNUM
1138  14     1:3     44       ELSE
1139  14     1:4     48         BEGIN
1140  14     1:5     48           IF ERRNUM<=109 THEN BLK:=2
1141  14     1:5     55           ELSE
1142  14     1:6     60             IF ERRNUM<=131 THEN BLK:=4
1143  14     1:6     69             ELSE
1144  14     1:7     74               IF ERRNUM<=156 THEN BLK:=6
1145  14     1:7     83               ELSE
1146  14     1:8     88                 IF ERRNUM<=254 THEN BLK:=8
1147  14     1:8     97                 ELSE BLK:=10;
1148  14     1:5    105           IF BLOCKREAD(F,BUF,2,BLK)<>2 THEN PUTNUM
1149  14     1:5    123           ELSE
```

```
1150  14    1:6    127              BEGIN
1151  14    1:7    127                IF BUF[0]=CHR(DLE) THEN PTR:=2 ELSE PTR:=0;
1152  14    1:7    143                D0:=ERRNUM DIV 100; (* CONVERT ERROR NUMBER TO CHARACTERS *)
1153  14    1:7    150                D1:=(ERRNUM-D0*100) DIV 10;
1154  14    1:7    161                D2:=ERRNUM MOD 10;
1155  14    1:7    168                T[0]:=CHR(D0+ORD('0')); T[1]:=CHR(D1+ORD('0'));
1156  14    1:7    182                T[2]:=CHR(D2+ORD('0'));
1157  14    1:7    189                REPEAT
1158  14    1:8    189                  FILLCHAR(C,3,'0');
1159  14    1:8    196                  COLON:=SCAN(MAXCHAR,=':',BUF[PTR]);
1160  14    1:8    209                  MOVELEFT(BUF[PTR],C[3-COLON],COLON);
1161  14    1:8    220                  COLON:=COLON+PTR;
1162  14    1:8    225                  PTR:=SCAN(MAXCHAR,=CHR(EOL),BUF[PTR])+PTR+3
1163  14    1:7    238                UNTIL (T=C) OR (BUF[PTR]=CHR(0));
1164  14    1:7    258                IF BUF[PTR]=CHR(0) THEN PUTNUM
1165  14    1:7    266                ELSE
1166  14    1:8    270                  BEGIN
1167  14    1:9    270                    MOVELEFT(BUF[COLON+1],MSG[1],(PTR-COLON)-4);
1168  14    1:9    286                    MSG[0]:=CHR(MIN(68,(PTR-COLON)-4)); (* R- REQUIRED *)
1169  14    1:9    302                    HOME; CLEARLINE(0); WRITE(MSG,'.  TYPE <SP>');
1170  14    1:8    341                  END
1171  14    1:6    341              END
1172  14    1:4    341            END(* IF IORESULT<>0 *);
1173  14    1:3    341          SHOWCURSOR;
1174  14    1:3    344          REPEAT UNTIL GETCH=' ';
1175  14    1:3    353          ERRBLK:=0; ERRSYM:=0; ERRNUM:=0; (*  ONLY YELL ONCE!!! *)
1176  14    1:2    365        END(* WITH USERINFO *)
1177  14    1:0    365 END(* PUTSYNTAX *);
1178  14    1:0    392
1179  14    1:0    392
1180  14    1:0    392 (*$TE D I T C O R E  -  BASIC COMMANDS*)
1181  14    1:0    392
1182  15    1:D      1 SEGMENT PROCEDURE EDITCORE;
1183  15    1:D      1
1184  15    1:D      1 (* CORE PROCEDURES.  EXECUTE THESE COMMANDS UNTIL EITHER A SET ENVIRONMENT
1185  15    1:D      1    COMES ALONG OR A QUIT COMMAND. *)
1186  15    1:D      1
1187  15    1:D      1
1188  15    1:D      1
1189  15    2:D      1 PROCEDURE NEXTCOMMAND; FORWARD;
1190  15    2:D      1
```

```
1191  15    3:0     1 PROCEDURE FIXDIRECTION;
1192  15    3:0     0 BEGIN
1193  15    3:1     0   IF COMMAND=FORWARDC THEN DIRECTION:='>' ELSE DIRECTION:='<';
1194  15    3:1    13   HOME; WRITE(DIRECTION); (* UPDATE PROMPT LINE *)
1195  15    3:1    24   SHOWCURSOR; NEXTCOMMAND
1196  15    3:0    27 END;
1197  15    3:0    42
1198  15    4:0     1 PROCEDURE BANISH;
1199  15    4:0     1 VAR
1200  15    4:0     1  CH: CHAR;
1201  15    4:0     0 BEGIN
1202  15    4:1     0   PROMPTLINE:=' BANISH: TO THE L(EFT OR RIGHT  <ESC>';
1203  15    4:1    45   PROMPT; NEEDPROMPT:=TRUE;
1204  15    4:1    52   SHOWCURSOR;
1205  15    4:1    55   REPEAT CH:=UCLC(GETCH) UNTIL CH IN ['L','R',CHR(ESC)];
1206  15    4:1    91   IF CH<>CHR(ESC) THEN BEGIN GOTOXY(7,0); ERASETOEOL(7,0) END;
1207  15    4:1   108   IF CH='L' THEN
1208  15    4:2   113     PUTPAGES(LEFTSTACK)
1209  15    4:1   114   ELSE
1210  15    4:2   119     IF CH='R' THEN
1211  15    4:3   124       PUTPAGES(RIGHTSTACK);
1212  15    4:1   128   IF CH<>CHR(ESC) THEN CENTERCURSOR(TRASH,MIDDLE,TRUE);
1213  15    4:1   145   NEXTCOMMAND
1214  15    4:0   145 END;
1215  15    4:0   162
1216  15    5:0     1 PROCEDURE NEXT;
1217  15    5:0     1 VAR
1218  15    5:0     1   CH: CHAR;
1219  15    5:0     0 BEGIN
1220  15    5:1     0   PROMPTLINE:=
1221  15    5:1     3 ' NEXT: F(ORWARDS, B(ACKWARDS IN THE FILE; S(TART, E(ND OF THE FILE.  <ESC>';
1222  15    5:1    82   PROMPT; NEEDPROMPT:=TRUE;
1223  15    5:1    89   SHOWCURSOR;
1224  15    5:1    92   REPEAT CH:=UCLC(GETCH) UNTIL CH IN ['F','B','S','E',CHR(ESC)];
1225  15    5:1   129   IF CH<>CHR(ESC) THEN BEGIN GOTOXY(5,0); ERASETOEOL(5,0) END;
1226  15    5:1   146   IF CH='F' THEN
1227  15    5:2   151     BEGIN
1228  15    5:3   151       PUTPAGES(LEFTSTACK);
1229  15    5:3   155       GETPAGES(RIGHTSTACK)
1230  15    5:2   156     END
1231  15    5:1   159   ELSE
```

```
1232  15    5:2   161      IF CH='B' THEN
1233  15    5:3   166        BEGIN
1234  15    5:4   166          PUTPAGES(RIGHTSTACK);
1235  15    5:4   170          GETPAGES(LEFTSTACK)
1236  15    5:3   171        END
1237  15    5:2   174      ELSE
1238  15    5:3   176        IF CH='S' THEN
1239  15    5:4   181          BEGIN
1240  15    5:5   181            WHILE LPAGE>0 DO
1241  15    5:6   188              BEGIN
1242  15    5:7   188                GOTOXY(5,0); ERASETOEOL(5,0);
1243  15    5:7   198                CURSOR:=1;
1244  15    5:7   201                PUTPAGES(RIGHTSTACK);
1245  15    5:7   205                GETPAGES(LEFTSTACK)
1246  15    5:6   206              END;
1247  15    5:5   211            CURSOR:=1
1248  15    5:4   211          END
1249  15    5:3   214        ELSE
1250  15    5:4   216          IF CH='E' THEN
1251  15    5:5   221            BEGIN
1252  15    5:6   221              WHILE RPAGE<FLENGTH DO
1253  15    5:7   230                BEGIN
1254  15    5:8   230                  GOTOXY(5,0); ERASETOEOL(5,0);
1255  15    5:8   240                  CURSOR:=BUFCOUNT-1;
1256  15    5:8   245                  PUTPAGES(LEFTSTACK);
1257  15    5:8   249                  GETPAGES(RIGHTSTACK)
1258  15    5:7   250                END;
1259  15    5:6   255              CURSOR:=BUFCOUNT-1;
1260  15    5:5   260            END;
1261  15    5:1   260    IF CH<>CHR(ESC) THEN CENTERCURSOR(TRASH,MIDDLE,TRUE);
1262  15    5:1   277    NEXTCOMMAND
1263  15    5:0   277  END;
1264  15    5:0   298
1265  15    6:D     1  PROCEDURE COPY;
1266  15    6:0     0  BEGIN
1267  15    6:1     0    PROMPTLINE:=' COPY: B(UFFER  F(ROM FILE  <ESC>';
1268  15    6:1    41    PROMPT; NEEDPROMPT:=TRUE;
1269  15    6:1    48    REPEAT
1270  15    6:2    48      CH:=UCLC(GETCH);
1271  15    6:1    60    UNTIL CH IN ['B','F',CHR(ESC)];
1272  15    6:1    83    IF CH='B' THEN
```

```
1273  15     6:2     38     BEGIN
1274  15     6:3     68       IF NOT COPYOK OR ((BUFCOUNT+COPYLENGTH+10>COPYSTART)
1275  15     6:3    103                         AND (COPYSTART>=BUFCOUNT))
1276  15     6:3    109         THEN ERROR('INVALID COPY.',NONFATAL)
1277  15     6:3    129         ELSE
1278  15     6:4    134           IF BUFCOUNT+COPYLENGTH>=BUFSIZE THEN ERROR('NO ROOM',NONFATAL)
1279  15     6:4    154           ELSE
1280  15     6:5    159             BEGIN
1281  15     6:6    159               IF COPYLINE THEN
1282  15     6:7    164                 BEGIN
1283  15     6:8    164                   GETLEADING;
1284  15     6:8    167                   CURSOR:=LINESTART
1285  15     6:7    167                 END;
1286  15     6:6    170               MOVERIGHT(EBUF^[CURSOR],EBUF^[CURSOR+COPYLENGTH],BUFCOUNT-CURSOR+1);
1287  15     6:6    185               IF (COPYSTART>=CURSOR) AND (COPYSTART<BUFCOUNT) THEN
1288  15     6:7    198                 MOVELEFT(EBUF^[COPYSTART+COPYLENGTH],EBUF^[CURSOR],COPYLENGTH)
1289  15     6:6    213               ELSE
1290  15     6:7    215                 MOVELEFT(EBUF^[COPYSTART],EBUF^[CURSOR],COPYLENGTH);
1291  15     6:6    226               BUFCOUNT:=BUFCOUNT+COPYLENGTH;
1292  15     6:6    233               READJUST(CURSOR,COPYLENGTH);
1293  15     6:6    240               CHECKINDENT(CURSOR);       (* CHECK THE BORDER FOR TWO DLE'S *)
1294  15     6:6    245               LASTPAT:=CURSOR;           (* FOR EQUALC *)
1295  15     6:6    248               CURSOR:=CURSOR+COPYLENGTH;
1296  15     6:6    255               CHECKINDENT(CURSOR);  (* ... AND ALSO CHECK THE OTHER BORDER *)
1297  15     6:6    260               GETLEADING;
1298  15     6:6    263               CURSOR:=MAX(CURSOR,STUFFSTART);
1299  15     6:6    272               CENTERCURSOR(TRASH,MIDDLE,TRUE)
1300  15     6:5    279             END;
1301  15     6:2    282       END (* CH='B' *)
1302  15     6:1    282     ELSE
1303  15     6:2    284       IF CH='F' THEN EXIT(EDITCORE);
1304  15     6:1    293     SHOWCURSOR;
1305  15     6:1    296     NEXTCOMMAND;
1306  15     6:0    298   END(*COPY*);
1307  15     6:0    316
1308  15     7:0      1   PROCEDURE DUMP;
1309  15     7:0      0   BEGIN
1310  15     7:1      0     NEXTCOMMAND;
1311  15     7:0      2   END(* DUMP *);
1312  15     7:0     14
1313  15     8:0      1   PROCEDURE FIND; FORWARD;
```

```
1314  15     8:0      1
1315  15     9:0      1 PROCEDURE INSERTIT; FORWARD;
1316  15     9:0      1
1317  15    10:0      1 PROCEDURE JUMP;
1318  15    10:0      1 VAR CH: CHAR;
1319  15    10:0      2
1320  15    11:0      1 PROCEDURE JUMPMARKER;
1321  15    11:0      1 VAR
1322  15    11:0      1   MUSTREDISP: BOOLEAN;
1323  15    11:0      2   I: INTEGER;
1324  15    11:0      3   MNAME: PACKED ARRAY [0..7] OF CHAR;
1325  15    11:0      7
1326  15    12:0      1 PROCEDURE SHUFFLE;
1327  15    12:0      0 BEGIN
1328  15    12:0      0   (* MUST REDISPLAY THE SCREEN *)
1329  15    12:1      0   MUSTREDISP:=TRUE;
1330  15    12:1      4   WITH PAGEZERO DO
1331  15    12:2      4     BEGIN
1332  15    12:3      4       CLEARLINE(0);
1333  15    12:3      8       WRITE('LEAPING');
1334  15    12:3     25       IF PAGEN[I]<=LPAGE THEN
1335  15    12:4     40         WHILE (LPAGE>0) AND (PAGEN[I]<>-1) DO
1336  15    12:5     60           BEGIN
1337  15    12:6     60             GOTOXY(7,0); ERASETOEOL(7,0);
1338  15    12:6     70             CURSOR:=1;
1339  15    12:6     73             PUTPAGES(RIGHTSTACK);
1340  15    12:6     77             GETPAGES(LEFTSTACK)
1341  15    12:5     78           END
1342  15    12:3     81       ELSE
1343  15    12:4     85         WHILE (RPAGE<FLENGTH) AND (PAGEN[I]<>-1) DO
1344  15    12:5    107           BEGIN
1345  15    12:6    107             GOTOXY(7,0); ERASETOEOL(7,0);
1346  15    12:6    117             CURSOR:=BUFCOUNT-1;
1347  15    12:6    122             PUTPAGES(LEFTSTACK);
1348  15    12:6    126             GETPAGES(RIGHTSTACK)
1349  15    12:5    127           END
1350  15    12:2    130     END
1351  15    12:0    132 END;
1352  15    12:0    148
1353  15    11:0      0 BEGIN
1354  15    11:1      0   MUSTREDISP:=FALSE;
```

```
1355  15   11:1     3    WITH PAGEZERO DO
1356  15   11:2     3      BEGIN
1357  15   11:3     3        GETNAME('JUMP TO',MNAME);
1358  15   11:3    18        IF MNAME<>'          ' THEN
1359  15   11:4    36          BEGIN
1360  15   11:5    36            I:=0;
1361  15   11:5    39            WHILE (I<COUNT) AND (MNAME<>NAME[I]) DO I:=I+1;
1362  15   11:5    65            IF MNAME<>NAME[I] THEN
1363  15   11:6    78              ERROR('NOT THERE.',NONFATAL)
1364  15   11:5    92            ELSE
1365  15   11:6    97              BEGIN
1366  15   11:6    97                (* IF TEXT POINTED TO ISN'T IN THE BUFFER, LOAD IT IN *)
1367  15   11:7    97                IF PAGEN[I]<>-1 THEN SHUFFLE;
1368  15   11:7   111                IF PAGEN[I]<>-1 THEN ERROR('MARKER ALL MESSED UP.',NONFATAL)
1369  15   11:7   148                ELSE
1370  15   11:8   153                  CURSOR:=POFFSET[I];
1371  15   11:7   162                GETLEADING;
1372  15   11:7   165                CURSOR:=MAX(CURSOR,STUFFSTART);
1373  15   11:7   174                CENTERCURSOR(TRASH,MIDDLE,MUSTREDISP)
1374  15   11:6   181              END;
1375  15   11:4   184          END;
1376  15   11:2   184      END;
1377  15   11:0   184 END; (* JUMPMARKER *)
1378  15   11:0   200
1379  15   10:0     0 BEGIN (* JUMP *)
1380  15   10:1     0   PROMPTLINE:=' JUMP: B(EGINNING E(ND M(ARKER <ESC>';
1381  15   10:1    44   PROMPT;
1382  15   10:1    47   NEEDPROMPT:=TRUE;  (* NEED TO REDISPLAY EDIT: PROMPTLINE! *)
1383  15   10:1    51   REPEAT
1384  15   10:2    51     CH:=UCLC(GETCH);
1385  15   10:2    63     IF CH='B' THEN
1386  15   10:3    68       BEGIN
1387  15   10:4    68         CURSOR:=1;
1388  15   10:4    71         GETLEADING;
1389  15   10:4    74         CURSOR:=STUFFSTART;
1390  15   10:4    77         CENTERCURSOR(TRASH,1,FALSE)
1391  15   10:3    82       END
1392  15   10:2    85     ELSE
1393  15   10:3    87       IF CH='E' THEN
1394  15   10:4    92         BEGIN
1395  15   10:5    92           CURSOR:=BUFCOUNT-1;
```

```
1396  15   10:5    97            CENTERCURSOR(TRASH,SCREENHEIGHT-1,FALSE);
1397  15   10:4   107          END
1398  15   10:3   107        ELSE
1399  15   10:4   109          IF CH='M' THEN JUMPMARKER
1400  15   10:4   114          ELSE IF CH<>CHR(ESC) THEN ERRWAIT;
1401  15   10:1   128    UNTIL (CH IN ['B','E','M',CHR(ESC)]);
1402  15   10:1   151    NEXTCOMMAND;
1403  15   10:0   153  END;
1404  15   10:0   168
1405  15   13:0     1  PROCEDURE DEFMACRO;
1406  15   13:0     0  BEGIN
1407  15   13:1     0    WITH PAGEZERO DO IF FILLING AND NOT AUTOINDENT THEN
1408  15   13:3    10      BEGIN
1409  15   13:4    10        BLANKCRT(1);
1410  15   13:4    14        THEFIXER(CURSOR,REPEATFACTOR,TRUE);
1411  15   13:4    20        CENTERCURSOR(TRASH,MIDDLE,TRUE);
1412  15   13:3    30      END
1413  15   13:2    30    ELSE ERROR('INAPPROPRIATE ENVIRONMENT',NONFATAL);
1414  15   13:1    64    COPYOK:=FALSE;
1415  15   13:1    68    SHOWCURSOR;
1416  15   13:1    71    NEXTCOMMAND;
1417  15   13:0    73  END;
1418  15   13:0    86
1419  15   14:0     1  PROCEDURE SETMARKER;
1420  15   14:0     1  LABEL 1;
1421  15   14:0     1  VAR
1422  15   14:0     1    I,SLOT: INTEGER;
1423  15   14:0     3    MNAME: PACKED ARRAY [0..7] OF CHAR;
1424  15   14:0     0  BEGIN
1425  15   14:1     0    WITH PAGEZERO DO
1426  15   14:2     0      BEGIN
1427  15   14:3     0        NEEDPROMPT:=TRUE;
1428  15   14:3     4        COUNT:=MIN(20,COUNT);
1429  15   14:3    16        IF COUNT=20 THEN
1430  15   14:4    23          BEGIN
1431  15   14:5    23            BLANKCRT(1);
1432  15   14:5    27            FOR I:=0 TO COUNT-1 DO
1433  15   14:6    42              BEGIN
1434  15   14:7    42                WRITE(CHR(ORD('A')+I),') ',NAME[I],'    ');
1435  15   14:7    91                IF (I+1) MOD 4 = 0 THEN WRITELN;
1436  15   14:6   106              END;
```

```
1437  15    14:5    113            MSG:=
1438  15    14:5    116      'MARKER OVERFLOW.  WHICH ONE TO REPLACE? (TYPE IN THE LETTER OR <SP>) ';
1439  15    14:5    190            PUTMSG; CH:=UCLC(GETCH);
1440  15    14:5    205            CENTERCURSOR(TRASH,MIDDLE,TRUE);
1441  15    14:5    215            IF CH IN ['A'..'T'] THEN SLOT:=ORD(CH)-ORD('A')
1442  15    14:5    236            ELSE
1443  15    14:6    241              GOTO 1;
1444  15    14:4    243          END
1445  15    14:3    243        ELSE
1446  15    14:4    245          SLOT:=COUNT;
1447  15    14:3    250        GETNAME('SET',MNAME);
1448  15    14:3    261        IF MNAME<>'        ' THEN
1449  15    14:4    279          BEGIN
1450  15    14:5    279            FOR I:=0 TO COUNT-1 DO
1451  15    14:6    294              IF NAME[I]=MNAME THEN SLOT:=I;
1452  15    14:5    317            NAME[SLOT]:=MNAME;
1453  15    14:5    327            POFFSET[SLOT]:=CURSOR;
1454  15    14:5    335            PAGEN[SLOT]:=-1;
1455  15    14:5    344            IF SLOT=COUNT THEN COUNT:=COUNT+1
1456  15    14:4    354          END;
1457  15    14:2    359      END;
1458  15    14:1    359 1:END;
1459  15    14:1    378
1460  15    15:D      1 PROCEDURE SETSTUFF;
1461  15    15:D      1 VAR CH: CHAR;
1462  15    15:0      0 BEGIN
1463  15    15:1      0   PROMPTLINE:=' SET: E(NVIRONMENT M(ARKER <ESC>';
1464  15    15:1     40   PROMPT; NEEDPROMPT:=TRUE;
1465  15    15:1     47   REPEAT
1466  15    15:2     47     CH:=UCLC(GETCH);
1467  15    15:2     59     IF CH='E' THEN EXIT(EDITCORE)
1468  15    15:2     68     ELSE
1469  15    15:3     70       IF CH='M' THEN SETMARKER
1470  15    15:3     75       ELSE IF CH<>CHR(ESC) THEN ERRWAIT;
1471  15    15:1     89   UNTIL CH IN ['E','M',CHR(ESC)];
1472  15    15:1    111   SHOWCURSOR;
1473  15    15:1    114   NEXTCOMMAND;
1474  15    15:0    116 END(* SETSTUFF *);
1475  15    15:0    130
1476  15    16:D      1 PROCEDURE VERIFY;
1477  15    16:0      0 BEGIN
```

```
1478  15   16:1     0   CENTERCURSOR(TRASH,MIDDLE,TRUE);
1479  15   16:1    10   SHOWCURSOR;
1480  15   16:1    13   NEXTCOMMAND
1481  15   16:0    13 END (* VERIFY *);
1482  15   16:0    28
1483  15   17:0     1 PROCEDURE XMACRO;
1484  15   17:0     1 VAR
1485  15   17:0     1   SAVEC,I: INTEGER;
1486  15   17:0     3   SAVE:PACKED ARRAY [0..MAXSTRING] OF CHAR;
1487  15   17:0     0 BEGIN
1488  15   17:1     0   PROMPTLINE:=' EXCHANGE: TEXT [<BS> A CHAR] [<ESC> ESCAPES; <ETX> ACCEPTS]';
1489  15   17:1    68   PROMPT; NEEDPROMPT:=TRUE;
1490  15   17:1    75   SHOWCURSOR;
1491  15   17:1    78   SAVEC:=CURSOR;
1492  15   17:1    81   I:=0;
1493  15   17:1    84   REPEAT
1494  15   17:2    84     CH:=GETCH;
1495  15   17:2    91     IF MAPTOCOMMAND(CH)=LEFT THEN
1496  15   17:3   101       BEGIN
1497  15   17:4   101         IF (CURSOR>SAVEC) THEN
1498  15   17:5   106           BEGIN
1499  15   17:6   106             I:=I-1; CURSOR:=CURSOR-1; (* DECREMENT BOTH PTRS *)
1500  15   17:6   116             EBUF^[CURSOR]:=SAVE[I]; (* RESTORE BUFFER *)
1501  15   17:6   123             WRITE(CHR(BSPCE),EBUF^[CURSOR],CHR(BSPCE));
1502  15   17:5   149           END
1503  15   17:3   149       END
1504  15   17:2   149     ELSE
1505  15   17:3   151       IF CH=CHR(EOL) THEN BEGIN ERRWAIT; SHOWCURSOR END
1506  15   17:3   162       ELSE
1507  15   17:4   164         IF NOT (CH IN [CHR(ETX),CHR(ESC)]) AND (EBUF^[CURSOR]<>CHR(EOL)) THEN
1508  15   17:5   184           BEGIN
1509  15   17:6   184             IF NOT (CH IN [' '..'^']) THEN CH:='?';
1510  15   17:6   212             SAVE[I]:=EBUF^[CURSOR];
1511  15   17:6   219             EBUF^[CURSOR]:=CH;
1512  15   17:6   223             I:=I+1; CURSOR:=CURSOR+1;
1513  15   17:6   233             WRITE(CH)
1514  15   17:5   241           END;
1515  15   17:1   241   UNTIL CH IN [CHR(ETX),CHR(ESC)];
1516  15   17:1   254   IF CH=CHR(ESC) THEN
1517  15   17:2   261     BEGIN
1518  15   17:3   261       CURSOR:=SAVEC;
```

```
1519  15    17:3    264        MOVELEFT(SAVE[0],EBUF^[CURSOR],I);
1520  15    17:3    272        SHOWCURSOR; WRITE(SAVE:I); SHOWCURSOR
1521  15    17:2    287      END;
1522  15    17:1    290    NEXTCOMMAND;
1523  15    17:0    292  END (* XMACRO *);
1524  15    17:0    306
1525  15    18:0      1  PROCEDURE ZAPIT;
1526  15    18:0      0  BEGIN
1527  15    18:1      0    IF ABS(LASTPAT-CURSOR)>80 THEN
1528  15    18:2      8      BEGIN
1529  15    18:3      8        PROMPTLINE:=
1530  15    18:3     11  ' WARNING! YOU ARE ABOUT TO ZAP MORE THAN 80 CHARS, DO YOU WISH TO ZAP? (Y/N)';
1531  15    18:3     92        PROMPT;
1532  15    18:3     95        NEEDPROMPT:=TRUE;
1533  15    18:3     99        IF UCLC(GETCH)<>'Y' THEN
1534  15    18:4    113          BEGIN
1535  15    18:5    113            SHOWCURSOR;
1536  15    18:5    116            NEXTCOMMAND;
1537  15    18:5    118            EXIT(ZAPIT)
1538  15    18:4    122          END;
1539  15    18:2    122      END;
1540  15    18:1    122    IF OKTODEL(MIN(CURSOR,LASTPAT),MAX(CURSOR,LASTPAT)) THEN
1541  15    18:2    143      BEGIN
1542  15    18:3    143        COPYLINE:=FALSE;
1543  15    18:3    147        READJUST(MIN(CURSOR,LASTPAT),-ABS(CURSOR-LASTPAT));
1544  15    18:3    162        IF CURSOR>LASTPAT THEN
1545  15    18:4    167          MOVELEFT(EBUF^[CURSOR],EBUF^[LASTPAT],BUFCOUNT-CURSOR)
1546  15    18:3    176        ELSE
1547  15    18:4    178          MOVELEFT(EBUF^[LASTPAT],EBUF^[CURSOR],BUFCOUNT-LASTPAT);
1548  15    18:3    187        BUFCOUNT:=BUFCOUNT-ABS(CURSOR-LASTPAT);
1549  15    18:3    195        CURSOR:=LASTPAT;
1550  15    18:3    198        CENTERCURSOR(TRASH,MIDDLE,TRUE);
1551  15    18:2    208      END;
1552  15    18:1    208    SHOWCURSOR;
1553  15    18:1    211    NEXTCOMMAND;
1554  15    18:0    213  END;
1555  15    18:0    226
1556  15    18:0    226  (*$TI N S E R T   C O M M A N D*)
1557  15    18:0    226
1558  15     9:0      1  PROCEDURE INSERTIT;
1559  15     9:0      1  CONST
```

```
1560  15     9:0      1   FUDGEFACTOR=10;
1561  15     9:0      1 VAR
1562  15     9:0      1   THEREST,LEFTPART,SAVEBUFCOUNT: PTRTYPE;
1563  15     9:0      4   CLEARED,WARNED,OK,NOTEXTYET,EXITPROMPT,FIRSTLINE: BOOLEAN;
1564  15     9:0     10   SPACES,LMOVE,X,LINE,EOLDIST,RJUST: INTEGER;
1565  15     9:0     16   CONTEXT: PACKED ARRAY [0..MAXSTRING] OF CHAR;
1566  15     9:0     80
1567  15    19:0      1 PROCEDURE SLAMRIGHT;
1568  15    19:0      1 (* MOVE (SLAM) THE PORTION OF THE EBUF^ TO THE RIGHT OF (AND INCLUDING)
1569  15    19:0      1    THE CURSOR SO THAT THE LAST NUL IN THE FILE (EBUF^[BUFCOUNT]) IS NOW AT
1570  15    19:0      1    EBUF^[BUFSIZE].  THEREST POINTS TO THE BEGINNING OF THE RIGHT-JUSTIFIED
1571  15    19:0      1    TEXT. *)
1572  15    19:0      0 BEGIN
1573  15    19:1      0   GETLEADING;
1574  15    19:1      3   THEREST:=BUFSIZE-(BUFCOUNT-CURSOR);
1575  15    19:1     11   LMOVE:=BUFCOUNT-CURSOR+1;
1576  15    19:1     19   MOVERIGHT(EBUF^[CURSOR],EBUF^[THEREST],LMOVE);
1577  15    19:1     30   GETLEADING; (* SET BLANKS *)
1578  15    19:1     33   IF THEREST-CURSOR<MAXSTRING THEN
1579  15    19:2     42     BEGIN
1580  15    19:3     42       ERROR('NO ROOM TO INSERT.',NONFATAL);
1581  15    19:3     67       SHOWCURSOR;
1582  15    19:3     70       NEXTCOMMAND;
1583  15    19:3     72       EXIT(INSERTIT)
1584  15    19:2     76     END;
1585  15    19:2     76   (* OPTIONAL INDENTATION *)
1586  15    19:1     76   EBUF^[THEREST-2]:=CHR(DLE); EBUF^[THEREST-1]:=CHR(BLANKS+32);
1587  15    19:0     94 END;
1588  15    19:0    106
1589  15    20:0      1 PROCEDURE WRAPUP;
1590  15    20:0      1 (* GIVEN THE NEW VALUE OF THE CURSOR (ONE PAST THE LAST VALID CHARACTER
1591  15    20:0      1    INSERTED INTO THE BUFFER), PUT BACK TOGETHER THE TWO HALVES OF THE
1592  15    20:0      1    BUFFER.  THEN, TO POLISH IT OFF, UPDATE THE SCREEN SO THAT THE REST OF
1593  15    20:0      1    THE EDITOR CAN COPE *)
1594  15    20:0      1 VAR PTR: PTRTYPE;
1595  15    20:0      2     LNGTH: INTEGER;
1596  15    20:0      0 BEGIN
1597  15    20:1      0   WITH PAGEZERO DO
1598  15    20:2      0     IF NOTEXTYET AND (NOT FIRSTLINE) AND
1599  15    20:2      8        ((NOT FILLING) OR AUTOINDENT) AND (CH<>CHR(ESC))
1600  15    20:2     22     THEN (* WE WANT THE BLANKS BEFORE THEREST *)
```

```
1601  15   20:3    25        BEGIN
1602  15   20:4    25          BUFCOUNT:=BUFCOUNT+2;
1603  15   20:4    30          THEREST:=THEREST-2; LMOVE:=LMOVE+2;
1604  15   20:4    46          CURSOR:=SCAN(-MAXCHAR,=CHR(EOL),EBUF^[CURSOR-1])+CURSOR;
1605  15   20:3    63        END;
1606  15   20:1    63    MOVELEFT(EBUF^[THEREST],EBUF^[CURSOR],LMOVE);
1607  15   20:1    74    READJUST(LEFTPART+1,CURSOR-(LEFTPART+1));
1608  15   20:1    89    BUFCOUNT:=BUFCOUNT+CURSOR-(LEFTPART+1);
1609  15   20:1   100    WITH PAGEZERO DO
1610  15   20:2   100      IF FILLING AND NOT AUTOINDENT AND (CH=CHR(ETX)) THEN
1611  15   20:3   116        BEGIN THEFIXER(CURSOR,1,FALSE); FIRSTLINE:=FALSE; FINDXY(X,LINE) END;
1612  15   20:1   135    UPSCREEN(FIRSTLINE,EXITPROMPT OR (CH=CHR(ESC)),LINE);
1613  15   20:1   153    GETLEADING;
1614  15   20:1   156    CURSOR:=MAX(CURSOR,STUFFSTART);
1615  15   20:1   165    LASTPAT:=LEFTPART+1;
1616  15   20:1   172    COPYOK:=TRUE; COPYSTART:=LASTPAT; COPYLENGTH:=CURSOR-LASTPAT;
1617  15   20:1   186    NEXTCOMMAND
1618  15   20:0   186  END;
1619  15   20:0   200
1620  15   21:0     3  FUNCTION CHECK(VALUE:INTEGER): BOOLEAN;
1621  15   21:0     4    (* VALUE IS THE POTENTIAL VALUE OF THE CURSOR.  IF IT IS NOT IN LEGAL
1622  15   21:0     4       RANGE THEN CHECK IS FALSE.  THIS FUNCTION ALSO WARNS THE USER IF
1623  15   21:0     4       S/HE IS GETTING TOO CLOSE TO OVERFLOWING THE BUFFER *)
1624  15   21:0     0  BEGIN
1625  15   21:1     0    CHECK:=TRUE;
1626  15   21:1     3    IF VALUE<=LEFTPART THEN
1627  15   21:2    10      BEGIN
1628  15   21:3    10        OK:=FALSE;  CHECK:=FALSE;
1629  15   21:3    17        ERROR('NO INSERTION TO BACK OVER.',NONFATAL); PROMPT;
1630  15   21:3    53        GOTOXY(X,LINE)
1631  15   21:2    62      END
1632  15   21:1    62    ELSE
1633  15   21:2    64      IF VALUE>=THEREST-MAXCHAR THEN
1634  15   21:3    75        BEGIN
1635  15   21:4    75          IF NOT WARNED THEN
1636  15   21:5    81            BEGIN
1637  15   21:6    81              ERROR('PLEASE FINISH UP THE INSERTION',NONFATAL); PROMPT;
1638  15   21:6   121              GOTOXY(X,LINE);
1639  15   21:6   130              WARNED:=TRUE
1640  15   21:5   130            END;
1641  15   21:4   134          IF VALUE>THEREST-FUDGE  CTOR THEN
```

```
1642  15   21:5   143            BEGIN
1643  15   21:6   143              ERROR('BUFFER OVERFLOW!!!!!',NONFATAL);
1644  15   21:6   169              WRAPUP;
1645  15   21:6   171              EXIT(INSERTIT);
1646  15   21:5   175            END
1647  15   21:3   175        END
1648  15   21:0   175 END;
1649  15   21:0   188
1650  15   22:0     1 PROCEDURE SPACEOVER;
1651  15   22:0     1 (* THIS PROCEDURE HANDLES SPACES AND TABS INSERTED INTO THE BUFFER *)
1652  15   22:0     1 VAR NEWX: INTEGER;
1653  15   22:0     0 BEGIN
1654  15   22:1     0   IF CH=CHR(HT) THEN
1655  15   22:2     5     BEGIN
1656  15   22:3     5       NEWX:=X+1;
1657  15   22:3    12       WITH PAGEZERO DO
1658  15   22:4    12         WHILE (TABSTOP[NEWX]=NONE) AND (NEWX<SCREENWIDTH) DO NEWX:=NEWX+1;
1659  15   22:3    35       SPACES:=NEWX-X
1660  15   22:2    36     END
1661  15   22:1    43   ELSE
1662  15   22:2    45     SPACES:=1;
1663  15   22:1    49   IF CHECK(CURSOR+SPACES) THEN
1664  15   22:2    60     BEGIN
1665  15   22:3    60       FILLCHAR(EBUF^[CURSOR],SPACES,' ');
1666  15   22:3    68       CURSOR:=CURSOR+SPACES
1667  15   22:2    69     END
1668  15   22:0    75 END;
1669  15   22:0    90
1670  15   23:0     1 PROCEDURE FIXUP; FORWARD;
1671  15   23:0     1
1672  15   24:0     1 PROCEDURE ENDLINE;
1673  15   24:0     1 (* FIRST, IF THERE WAS NO TEXT INSERTED ON THE CURRENT LINE, THEN CONVERT
1674  15   24:0     1    ALL OF THE SPACES TO BLANK COMPRESSION CODES.  THEN INSERT AN <EOL> INTO
1675  15   24:0     1    THE BUFFER FOLLOWED BY THE APPROPRIATE NUMBER OF SPACES FOR THE
1676  15   24:0     1    INDENTATION. *)
1677  15   24:0     0 BEGIN
1678  15   24:1     0   WITH PAGEZERO DO
1679  15   24:2     0     BEGIN
1680  15   24:3     0       IF NOTEXTYET THEN FIXUP;
1681  15   24:3     7       EBUF^[CURSOR]:=CHR(EOL);
1682  15   24:3    11       IF AUTOINDENT THEN GETLEADING
```

```
1683  15   24:3    16        ELSE
1684  15   24:4    21          IF FILLING THEN
1685  15   24:5    26            BEGIN
1686  15   24:6    26              GETLEADING;
1687  15   24:6    29              IF EBUF^[STUFFSTART]=CHR(EOL) THEN (* EMPTY LINE *)
1688  15   24:7    36                BLANKS:=PARAMARGIN
1689  15   24:6    36              ELSE BLANKS:=LMARGIN
1690  15   24:5    43            END
1691  15   24:4    48          ELSE BLANKS:=0;
1692  15   24:3    53        IF CHECK(CURSOR+BLANKS+1) THEN
1693  15   24:4    64          BEGIN
1694  15   24:5    64            FILLCHAR(EBUF^[CURSOR+1],BLANKS,' ');
1695  15   24:5    72            CURSOR:=CURSOR+BLANKS+1
1696  15   24:4    75          END;
1697  15   24:3    79        NOTEXTYET:=TRUE;
1698  15   24:2    83      END;
1699  15   24:0    83 END;
1700  15   24:0    96
1701  15   25:D     1 PROCEDURE BACKUP;
1702  15   25:D     1 (* IF THE CH IS A BACKSPACE THEN DECREMENT CURSOR BY 1.  IF THIS WOULD
1703  15   25:D     1    RESULT IN BACKING OVER AN <EOL> OR A BLANK COMPRESSION CODE, THEN FALL
1704  15   25:D     1    INTO THE CODE FOR A <DEL> (ALSO CHANGING THE CH TO <DEL> FOR COMMUNICATION
1705  15   25:D     1    TO THE OUTER BLOCK)  *)
1706  15   25:D     1 VAR PTR: PTRTYPE;
1707  15   25:0     0 BEGIN
1708  15   25:1     0   IF CH=CHR(DC1) THEN
1709  15   25:2     5     BEGIN GETLEADING; IF CHECK(LINESTART) THEN CURSOR:=LINESTART END
1710  15   25:1    18   ELSE
1711  15   25:2    20     IF (CH=CHR(BS)) AND
1712  15   25:2    25        NOT( (EBUF^[CURSOR-2]=CHR(DLE)) OR (EBUF^[CURSOR-1]=CHR(EOL)) ) THEN
1713  15   25:3    44       BEGIN
1714  15   25:4    44         IF CURSOR<LEFTPART+2 THEN OK:=FALSE ELSE CURSOR:=CURSOR-1;
1715  15   25:3    64       END
1716  15   25:2    64     ELSE
1717  15   25:3    66       BEGIN (* A <DEL> OR EQUIVALENT *)
1718  15   25:4    66         CH:=CHR(DEL); (* TELL THE CRT DRIVER THAT THE LINE HAS CHANGED *)
1719  15   25:4    71         GETLEADING;
1720  15   25:4    74         IF CHECK(LINESTART-1) THEN CURSOR:=LINESTART-1;
1721  15   25:4    88         NOTEXTYET:=FALSE; (* THANK YOU SHAWN! *)
1722  15   25:3    92       END
1723  15   25:0    92 END;
```

```
1724  15   25:0   104
1725  15   23:D     1 PROCEDURE FIXUP;
1726  15   23:D     1 (* CONVERT THE INDENTATION SPACES INTO BLANK COMPRESSION CODES, AND MOVE
1727  15   23:D     1    THE CURRENT LINE AROUND ACCORDINGLY *)
1728  15   23:0     0 BEGIN
1729  15   23:0     0   (* FIRST COMPRESS THE CURRENT LINE *)
1730  15   23:1     0   EBUF^[CURSOR]:=CHR(EOL); (* FOOL GETLEADING *)
1731  15   23:1     4   GETLEADING;
1732  15   23:1     7   IF BYTES >= 2 THEN (* OK TO PUT IN <DLE> # AS IT STANDS *)
1733  15   23:2    12     MOVELEFT(EBUF^[STUFFSTART],EBUF^[LINESTART+2],CURSOR-STUFFSTART)
1734  15   23:1    23   ELSE
1735  15   23:2    25     IF CHECK(CURSOR+2-BYTES) THEN
1736  15   23:3    36       MOVERIGHT(EBUF^[STUFFSTART],EBUF^[STUFFSTART+2-BYTES],CURSOR-STUFFSTART)
1737  15   23:2    49     ELSE BEGIN OK:=FALSE; EXIT(FIXUP) END;
1738  15   23:1    59   CURSOR:=CURSOR-(BYTES-2);
1739  15   23:1    66   EBUF^[LINESTART]:=CHR(DLE); EBUF^[LINESTART+1]:=CHR(32+BLANKS);
1740  15   23:0    78 END;
1741  15   23:0    90
1742  15   26:D     1 PROCEDURE INSERTCH;
1743  15   26:D     1   (* THIS PROCEDURE INSERTS A SINGLE CHARACTER INTO THE BUFFER. IT ALSO
1744  15   26:D     1      HANDLES ALL OF THE CONTROL CODES (EOL,BS,DEL) AND BUFFER OVER- AND
1745  15   26:D     1      UNDER- FLOW CONDITIONS.  INSERTCH IS CALLED BY THE CRT HANDLER *)
1746  15   26:0     0 BEGIN
1747  15   26:1     0   REPEAT
1748  15   26:2     0   OK:=TRUE; (* NO ERRORS THAT INVALIDATE THE CURRENT CHARACTER HAVE OCCURED *)
1749  15   26:2     4   CH:=GETCH;
1750  15   26:2    11   IF MAPTOCOMMAND(CH)=LEFT THEN CH:=CHR(BS);
1751  15   26:2    26   IF ORD(CH) IN [SP,HT,EOL,BS,DEL,ETX,ESC,DC1] THEN
1752  15   26:3    59     BEGIN
1753  15   26:3    59       (* <ETX> AND <ESC> ARE HANDLED IN THE BODY OF INSERTIT *)
1754  15   26:4    59       IF ORD(CH) IN [SP,HT] THEN SPACEOVER
1755  15   26:4    72       ELSE
1756  15   26:5    76         IF ORD(CH)=EOL THEN ENDLINE
1757  15   26:5    81         ELSE
1758  15   26:6    85           IF ORD(CH) IN [DC1,BS,DEL] THEN BACKUP;
1759  15   26:3   109     END
1760  15   26:2   109   ELSE
1761  15   26:3   111     BEGIN (* A CHARACTER TO INSERT! *)
1762  15   26:4   111       IF (CH<'!') OR (CH>'^') THEN CH:='?'; (* NO NON-PRINTING CHARACTERS *)
1763  15   26:4   123       IF NOTEXTYET THEN FIXUP;
1764  15   26:4   130       IF CHECK(CURSOR+1) AND OK THEN
```

```
1765  15   26:5   143          BEGIN
1766  15   26:6   143            NOTEXTYET:=FALSE;
1767  15   26:6   147            EBUF^[CURSOR]:=CH;
1768  15   26:6   151            CURSOR:=CURSOR+1
1769  15   26:5   152          END;
1770  15   26:3   156      END;
1771  15   26:1   156  UNTIL OK;
1772  15   26:0   161 END;
1773  15   26:0   176
1774  15   27:0     1 PROCEDURE POPDOWN;
1775  15   27:0     1 (* DISPLAYS CONTEXT, DOING AN IMPLIED SCROLLUP IF NEC. *)
1776  15   27:0     0 BEGIN
1777  15   27:1     0   IF CLEARED THEN ERASETOEOL(X,LINE)
1778  15   27:1    11   ELSE BEGIN CLEARED:=TRUE; ERASEOS(X,LINE) END;
1779  15   27:1    29   GOTOXY(RJUST,LINE);
1780  15   27:1    38   ERASETOEOL(RJUST,LINE);
1781  15   27:1    47   WRITE(CHR(LF));
1782  15   27:1    55   IF LINE=SCREENHEIGHT THEN BEGIN EXITPROMPT:=TRUE; LINE:=SCREENHEIGHT-1 END;
1783  15   27:1    72   WRITE(CONTEXT:EOLDIST);
1784  15   27:1    87   FIRSTLINE:=FALSE; (* SAYS THAT THE WHOLE SCREEN HAS BEEN AFFECTED. *)
1785  15   27:0    91 END;
1786  15   27:0   104
1787  15   28:0     1 PROCEDURE WRITESP(CH:CHAR;HOWMANY:INTEGER);
1788  15   28:0     0 BEGIN
1789  15   28:1     0   IF X+HOWMANY<=SCREENWIDTH THEN WRITE(CH:HOWMANY);
1790  15   28:1    17   IF X+HOWMANY>=SCREENWIDTH THEN
1791  15   28:2    26     BEGIN
1792  15   28:3    26       GOTOXY(SCREENWIDTH,LINE);
1793  15   28:3    33       IF X+HOWMANY>SCREENWIDTH THEN
1794  15   28:4    42         BEGIN WRITE('!'); GOTOXY(SCREENWIDTH,LINE) END
1795  15   28:2    57     END;
1796  15   28:1    57   X:=MIN(SCREENWIDTH,X+HOWMANY)
1797  15   28:0    63 END;
1798  15   28:0    84
1799  15   29:0     1 PROCEDURE CLEANSCREEN;
1800  15   29:0     1 (* CODE TO, IF POSSIBLE, ONLY ERASE THE LINE, OTHERWISE CLEAR
1801  15   29:0     1    THE SCREEN.  THEN CALL POPDOWN *)
1802  15   29:0     0 BEGIN
1803  15   29:1     0   FIRSTLINE:=FALSE;
1804  15   29:1     4   IF CLEARED THEN
1805  15   29:2     9     BEGIN
```

```
1806  15    29:3      9        IF X<SCREENWIDTH THEN ERASETOEOL(X,LINE)
1807  15    29:2     22      END
1808  15    29:1     25    ELSE
1809  15    29:2     27      BEGIN
1810  15    29:3     27        CLEARED:=TRUE; ERASEOS(X,LINE);
1811  15    29:2     40      END;
1812  15    29:1     40    LINE:=LINE+1;
1813  15    29:1     48    IF LINE>SCREENHEIGHT THEN
1814  15    29:2     55      BEGIN
1815  15    29:3     55        LINE:=LINE-1;
1816  15    29:3     63        WRITELN;
1817  15    29:3     69        EXITPROMPT:=TRUE
1818  15    29:2     69      END;
1819  15    29:1     73    IF EOLDIST<>0 THEN POPDOWN
1820  15    29:0     80  END;
1821  15    29:0     94
1822  15    30:0      1  PROCEDURE POPOV;
1823  15    30:0      1  (* WHEN IN FILLING MODE, THIS PROCEDURE IS CALLED WHEN A LINE IS OVERFLOWED
1824  15    30:0      1     (X >= RIGHTMARGIN).  THE WORD IS SCANNED OFF AND "POPPED" DOWN TO THE
1825  15    30:0      1     NEXT LINE. *)
1826  15    30:0      1  VAR
1827  15    30:0      1    WLENGTH: INTEGER;
1828  15    30:0      2    SAVE,PTR: PTRTYPE;
1829  15    30:0      4    WORD: PACKED ARRAY [0..MAXSW] OF CHAR;
1830  15    30:0      0  BEGIN
1831  15    30:1      0    IF NOTEXTYET THEN FIXUP;
1832  15    30:1      7    PTR:=MAX(SCAN(-MAXCHAR,='-',EBUF^[CURSOR-1]),
1833  15    30:1     20             SCAN(-MAXCHAR,=' ',EBUF^[CURSOR-1]))+CURSOR;
1834  15    30:1     42    WLENGTH:=CURSOR-PTR;
1835  15    30:1     47    WITH PAGEZERO DO IF WLENGTH>=RMARGIN-LMARGIN THEN
1836  15    30:3     58      BEGIN
1837  15    30:4     58        WRITESP(CH,1);
1838  15    30:4     62        EXIT(POPOV)
1839  15    30:3     66      END;
1840  15    30:1     66    IF CH='-' THEN WRITE('-');
1841  15    30:1     79    GOTOXY(X-WLENGTH+1,LINE);
1842  15    30:1     92    ERASETOEOL(X-WLENGTH+1,LINE);
1843  15    30:1    105    MOVERIGHT(EBUF^[PTR],EBUF^[PTR+3],WLENGTH);
1844  15    30:1    114    MOVELEFT(EBUF^[PTR+3],WORD,WLENGTH);
1845  15    30:1    124    CURSOR:=CURSOR+3;
1846  15    30:1    129    EBUF^[PTR]:=CHR(EOL);
```

```
1847  15   30:1   133   EBUF^[PTR+1]:=CHR(DLE);
1848  15   30:1   139   WITH PAGEZERO DO IF AUTOINDENT THEN
1849  15   30:3   144     BEGIN
1850  15   30:4   144       SAVE:=CURSOR; (* SET BLANKS TO THE INDENTATION OF THE LINE ABOVE *)
1851  15   30:4   147       CURSOR:=PTR;
1852  15   30:4   150       GETLEADING;
1853  15   30:4   153       CURSOR:=SAVE
1854  15   30:3   153     END
1855  15   30:2   156   ELSE
1856  15   30:3   158     BLANKS:=LMARGIN;
1857  15   30:1   163   EBUF^[PTR+2]:=CHR(BLANKS+32);
1858  15   30:1   171   CLEANSCREEN;
1859  15   30:1   173   X:=BLANKS;
1860  15   30:1   177   GOTOXY(X,LINE); WRITE(WORD:WLENGTH);
1861  15   30:1   196   X:=X+WLENGTH;
1862  15   30:1   204   NOTEXTYET:=FALSE
1863  15   30:0   204 END;
1864  15   30:0   220
1865  15    9:0     0 BEGIN (* INSERT *)
1866  15    9:1     0   CLEARED:=FALSE;
1867  15    9:1     3   EOLDIST:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR]);
1868  15    9:1    15   MOVELEFT(EBUF^[CURSOR],CONTEXT[0],EOLDIST);
1869  15    9:1    23   RJUST:=SCREENWIDTH-EOLDIST;
1870  15    9:1    28   SLAMRIGHT;
1871  15    9:1    30   SAVEBUFCOUNT:=BUFCOUNT;
1872  15    9:1    33   PROMPTLINE:=
1873  15    9:1    36 ' INSERT: TEXT [<BS> A CHAR,<DEL> A LINE]  [<ETX> ACCEPTS, <ESC> ESCAPES]';
1874  15    9:1   113   PROMPT;
1875  15    9:1   116   EXITPROMPT:=FALSE;  NEEDPROMPT:=TRUE;
1876  15    9:1   123   LEFTPART:=CURSOR-1;
1877  15    9:1   128   NOTEXTYET:=FALSE;
1878  15    9:1   131   FINDXY(X,LINE);  GOTOXY(X,LINE);
1879  15    9:1   143   ERASETOEOL(X,LINE);
1880  15    9:1   148   FIRSTLINE:=TRUE;
1881  15    9:1   151   IF EOLDIST<>0 THEN (* A CONTEXT NEEDS TO BE DISPLAYED *)
1882  15    9:2   156     IF RJUST>X THEN (* AND IT WILL FIT ON THE CURRENT LINE ... *)
1883  15    9:3   161       BEGIN
1884  15    9:4   161         GOTOXY(RJUST,LINE); WRITE(CONTEXT:EOLDIST); GOTOXY(X,LINE)
1885  15    9:3   183       END
1886  15    9:2   183     ELSE (* AND IT WON'T FIT ON THE CURRENT LINE *)
1887  15    9:3   185       BEGIN
```

```
1888  15    9:4   185          FIRSTLINE:=FALSE;
1889  15    9:4   188          ERASEOS(X,LINE);(* CLEAR THE SCREEN *)
1890  15    9:4   193          WRITELN;
1891  15    9:4   199          IF LINE=SCREENHEIGHT THEN
1892  15    9:5   204            BEGIN LINE:=SCREENHEIGHT-1; EXITPROMPT:=TRUE END;
1893  15    9:4   212          GOTOXY(RJUST,LINE+1); WRITE(CONTEXT:EOLDIST); GOTOXY(X,LINE)
1894  15    9:3   236        END;
1895  15    9:1   236    REPEAT
1896  15    9:2   236      INSERTCH;
1897  15    9:2   238      IF NOT (ORD(CH) IN [EOL,ETX,ESC,DEL,DC1]) THEN
1898  15    9:3   266        BEGIN
1899  15    9:4   266          IF TRANSLATE[CH]=LEFT THEN
1900  15    9:5   277            BEGIN IF X<=SCREENWIDTH THEN WRITE(CHR(BSPCE),' ',CHR(BSPCE)); X:=X-1 EN
1901  15    9:4   311          ELSE
1902  15    9:5   313            IF CH=CHR(HT) THEN WRITESP(' ',SPACES)
1903  15    9:5   320            ELSE
1904  15    9:6   324              IF PAGEZERO.FILLING AND (X+1>=PAGEZERO.RMARGIN) THEN POPOV
1905  15    9:6   337              ELSE WRITESP(CH,1);
1906  15    9:4   345          IF NOT PAGEZERO.FILLING AND (X=SCREENWIDTH-8) AND (CH<>CHR(BS))
1907  15    9:4   360            THEN WRITE(CHR(BELL));
1908  15    9:4   371          IF (EOLDIST<>0) AND
1909  15    9:4   374             (X>=RJUST) AND FIRSTLINE THEN  (*RAN INTO CONTEXT *)
1910  15    9:5   382            BEGIN
1911  15    9:6   382              POPDOWN;
1912  15    9:6   384              GOTOXY(X,LINE)
1913  15    9:5   389            END;
1914  15    9:3   389        END
1915  15    9:2   389      ELSE (* CH IN [EOL,ETX,ESC,DEL,DC1] *)
1916  15    9:3   391        BEGIN
1917  15    9:4   391          IF CH=CHR(EOL) THEN
1918  15    9:5   396            BEGIN
1919  15    9:6   396              CLEANSCREEN;
1920  15    9:6   398              X:=BLANKS;
1921  15    9:6   401              GOTOXY(X,LINE);
1922  15    9:5   406            END
1923  15    9:4   406          ELSE
1924  15    9:5   408            IF CH=CHR(DEL) THEN
1925  15    9:6   415              BEGIN
1926  15    9:7   415                IF LINE<=1 THEN  (* RUBBED OUT ALL OF WHAT WAS ON THE SCREEN *)
1927  15    9:8   420                  BEGIN
1928  15    9:9   420                    BUFCOUNT:=CURSOR+1;
```

```
1929  15    9:9   425                  EBUF^[CURSOR]:=CHR(EOL);
1930  15    9:9   429                  CENTERCURSOR(LINE,MIDDLE,TRUE);
1931  15    9:9   438                  IF EOLDIST<>0 THEN POPDOWN;
1932  15    9:9   445                  IF EXITPROMPT THEN BEGIN PROMPT; EXITPROMPT:=FALSE END
1933  15    9:8   454                END
1934  15    9:7   454              ELSE
1935  15    9:8   456                BEGIN GOTOXY(0,LINE); CLEARED:=FALSE;
1936  15    9:9   464                      ERASETOEOL(0,LINE); LINE:=LINE-1 END;
1937  15    9:7   474              GETLEADING;
1938  15    9:7   477              X:=BLANKS-BYTES+CURSOR-LINESTART;
1939  15    9:7   486              GOTOXY(X,LINE)
1940  15    9:6   491            END
1941  15    9:5   491          ELSE
1942  15    9:6   493            IF CH=CHR(DC1) THEN
1943  15    9:7   498              BEGIN
1944  15    9:8   498                X:=0; GOTOXY(X,LINE); ERASETOEOL(X,LINE)
1945  15    9:7   508              END;
1946  15    9:3   511        END;
1947  15    9:1   511    UNTIL CH IN [CHR(ETX),CHR(ESC)];
1948  15    9:1   524    IF CH=CHR(ESC) THEN CURSOR:=LEFTPART+1;
1949  15    9:1   536    BUFCOUNT:=SAVEBUFCOUNT;
1950  15    9:1   539    WRAPUP;
1951  15    9:0   541  END;
1952  15    9:0   556
1953  15    9:0   556  (*$TM O V E I T   -   CURSOR MOVEMENT, PAGE, ADJUST, DELETE *)
1954  15    9:0   556
1955  15   31:D     1  PROCEDURE MOVEIT;
1956  15   31:D     1  VAR
1957  15   31:D     1    SCROLLMARK,X,LINE,I: INTEGER;
1958  15   31:D     5    EXITPROMPT: BOOLEAN; (* PROMPT AFTER LEAVING MOVEIT! *)
1959  15   31:D     6    OLDLINE,OLDX: INTEGER;
1960  15   31:D     8    NEWDIST,DIST: INTEGER;
1961  15   31:D    10    DOFFSCREEN,ATEND,INREPLACE,INDELETE: BOOLEAN;
1962  15   31:D    14    PTR,ANCHOR,OLDCURSOR: PTRTYPE;
1963  15   31:D    17
1964  15   32:D     1  PROCEDURE SCROLLUP(BOTTOMLINE:PTRTYPE; HOWMANY: INTEGER);
1965  15   32:D     3  (* BOTTOMLINE IS THE "LINESTART" OF THE LINE TO BE SCROLLED UP *)
1966  15   32:D     3  VAR
1967  15   32:D     3    PTR: PTRTYPE;
1968  15   32:D     4    I: INTEGER;
1969  15   32:0     0  BEGIN
```

```
1970  15   32:0     0   (* DISPLAY THE NEXT LINE ON THE BOTTOM OF THE SCREEN *)
1971  15   32:1     0   I:=0;
1972  15   32:1     3   PTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[LINE1PTR])+LINE1PTR+1;
1973  15   32:1    23   WHILE (I<HOWMANY) AND (PTR<BUFCOUNT) DO
1974  15   32:2    32     BEGIN
1975  15   32:3    32       LINE1PTR:=PTR; PTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR])+PTR+1;
1976  15   32:3    52       I:=I+1
1977  15   32:2    53     END;
1978  15   32:1    59   I:=0;
1979  15   32:1    62   GOTOXY(0,SCREENHEIGHT);
1980  15   32:1    67   REPEAT
1981  15   32:2    67     I:=I+1;
1982  15   32:2    72     BLANKS:=LEADBLANKS(BOTTOMLINE,BYTES);
1983  15   32:2    82     WRITE(CHR(LF));
1984  15   32:2    90     LINEOUT(BOTTOMLINE,BYTES,BLANKS,SCREENHEIGHT);
1985  15   32:2    98     LINE:=LINE-1;
1986  15   32:1   106   UNTIL (I>=HOWMANY) OR (BOTTOMLINE>=BUFCOUNT-1);
1987  15   32:1   117   EXITPROMPT:=TRUE;
1988  15   32:0   121 END(* SCROLLUP *);
1989  15   32:0   138
1990  15   33:D     1 PROCEDURE CLEAR(X1,Y1,X2,Y2: INTEGER); FORWARD;
1991  15   33:D     5
1992  15   34:D     1 PROCEDURE CENTER;
1993  15   34:0     0 BEGIN
1994  15   34:1     0   IF INDELETE THEN
1995  15   34:2     5     BEGIN
1996  15   34:3     5       IF LINE>=SCREENHEIGHT THEN
1997  15   34:4    12         BEGIN
1998  15   34:5    12           CENTERCURSOR(LINE,2,TRUE);
1999  15   34:5    20           IF ABS(CURSOR-ANCHOR) > ABS(DIST) THEN CLEAR(0,1,MAX(X-1,0),LINE)
2000  15   34:4    49         END
2001  15   34:3    51       ELSE
2002  15   34:4    53         BEGIN
2003  15   34:5    53           CENTERCURSOR(LINE,SCREENHEIGHT-1,TRUE);
2004  15   34:5    63           GOTOXY(X,LINE);
2005  15   34:5    72           IF ABS(CURSOR-ANCHOR) > ABS(DIST) THEN WRITE(CHR(11))
2006  15   34:4    93         END;
2007  15   34:3    93       DOFFSCREEN:=TRUE;
2008  15   34:2    97     END
2009  15   34:1    97   ELSE
2010  15   34:2    99     IF (COMMAND=PARAC) AND ((DIRECTION='<') OR (LINE MOD SCREENHEIGHT=OLDLINE))
```

```
2011  15   34:2   115        THEN CENTERCURSOR(LINE,OLDLINE,TRUE)
2012  15   34:2   125        ELSE CENTERCURSOR(LINE,MIDDLE,TRUE);
2013  15   34:1   140    IF EXITPROMPT AND (COMMAND<>QUITC) THEN
2014  15   34:2   149      BEGIN
2015  15   34:3   149        PROMPT; EXITPROMPT:=FALSE
2016  15   34:2   152      END;
2017  15   34:1   156    OLDLINE:=LINE; OLDX:=X;
2018  15   34:0   168  END;
2019  15   34:0   180
2020  15   35:0     1  PROCEDURE UPMOVE;
2021  15   35:0     1  VAR I:INTEGER;
2022  15   35:0     0  BEGIN
2023  15   35:1     0    I:=1;
2024  15   35:1     3    GETLEADING;
2025  15   35:1     6    (* FIND THE LINE FIRST *)
2026  15   35:1     6    WHILE (I<=REPEATFACTOR) AND (LINESTART>1) DO
2027  15   35:2    15      BEGIN
2028  15   35:3    15        CURSOR:=LINESTART-1; (* LAST CHAR OF LINE ABOVE *)
2029  15   35:3    20        GETLEADING;
2030  15   35:3    23        LINE:=LINE-1;  I:=I+1;
2031  15   35:2    36      END;
2032  15   35:2    38    (* IF POSSIBLE SET THE CURSOR AT THE SAME X COORD WE CAME FROM.  OTHERWISE,
2033  15   35:2    38       SET IT EITHER TO THE BEGINNING OF THE BUFFER, THE BEGINNING OF TEXT
2034  15   35:2    38       ON THAT LINE, OR THE END OF THE TEXT ON THAT LINE *)
2035  15   35:1    38    CURSOR:=
2036  15   35:1    38              MAX(1,      (* THE BEGINNING OF THE BUFFER *)
2037  15   35:1    39                  MAX(STUFFSTART,  (* THE BEGINNING OF THE TEXT *)
2038  15   35:1    40                      MIN(X-BLANKS+BYTES+LINESTART, (* SAME COL *)
2039  15   35:1    49                          SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR (* EOL *)
2040  15   35:1    59                         )
2041  15   35:1    61                       )
2042  15   35:1    66                  );
2043  15   35:1    78    IF LINE<1 THEN CENTER;
2044  15   35:0    87  END(* UPALINE *);
2045  15   35:0   102
2046  15   36:0     1  PROCEDURE DOWNMOVE;
2047  15   36:0     1  VAR
2048  15   36:0     1    I: INTEGER;
2049  15   36:0     2    NEXTEOL: PTRTYPE;
2050  15   36:0     0  BEGIN
2051  15   36:1     0    I:=1;
```

```
2052  15   36:1    3   NEXTEOL:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR;
2053  15   36:1   17   WHILE (NEXTEOL<BUFCOUNT-1) AND (I<=REPEATFACTOR) DO
2054  15   36:2   28     BEGIN
2055  15   36:3   28       CURSOR:=NEXTEOL+1;
2056  15   36:3   33       NEXTEOL:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR;
2057  15   36:3   47       IF NEXTEOL<BUFCOUNT THEN
2058  15   36:4   52         BEGIN
2059  15   36:5   52           LINE:=LINE+1;
2060  15   36:5   60           I:=I+1;
2061  15   36:5   65           IF LINE=SCREENHEIGHT+1 THEN
2062  15   36:6   74             BEGIN
2063  15   36:7   74               SCROLLMARK:=CURSOR;
2064  15   36:6   78             END;
2065  15   36:4   78         END;
2066  15   36:2   78     END;
2067  15   36:1   80   IF LINE>SCREENHEIGHT THEN
2068  15   36:2   87     IF (LINE-SCREENHEIGHT>=SCREENHEIGHT) OR (INDELETE) THEN
2069  15   36:3  100       CENTER
2070  15   36:2  100     ELSE
2071  15   36:3  104       SCROLLUP(SCROLLMARK,LINE-SCREENHEIGHT);
2072  15   36:1  114   GETLEADING;
2073  15   36:1  117   (* IF POSSIBLE SET THE CURSOR AT THE SAME X COORD WE CAME FROM.  OTHERWISE,
2074  15   36:1  117      SET IT EITHER TO THE END OF THE BUFFER, THE BEGINNING OF TEXT
2075  15   36:1  117      ON THAT LINE, OR THE END OF THE TEXT ON THAT LINE *)
2076  15   36:1  117   CURSOR:=MIN(BUFCOUNT-1,       (* END OF THE BUFFER *)
2077  15   36:1  120               MAX(STUFFSTART,    (* NOT IN THE INDENTATION *)
2078  15   36:1  121                   MIN(X-BLANKS+BYTES+LINESTART (* WHERE IT WANTS TO BE *)
2079  15   36:1  128                      ,SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR
2080  15   36:1  140                      )
2081  15   36:1  142                   )
2082  15   36:1  147               );
2083  15   36:0  159 END(* DOWNMOVE *);
2084  15   36:0  174
2085  15   37:0    1 PROCEDURE LEFTMOVE;
2086  15   37:0    0 BEGIN
2087  15   37:1    0   GETLEADING; (* SET LINESTART AND STUFFSTART *)
2088  15   37:1    3   WHILE (STUFFSTART>CURSOR-REPEATFACTOR) AND (CURSOR>REPEATFACTOR) DO
2089  15   37:2   14     BEGIN
2090  15   37:3   14       REPEATFACTOR:=REPEATFACTOR-(CURSOR-STUFFSTART+1); (* CHARS MOVED OVER *)
2091  15   37:3   23       IF EBUF^[CURSOR]=CHR(EOL) THEN CURSOR:=CURSOR-1;
2092  15   37:3   35       CURSOR:=MAX(SCAN(-MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR,1);
```

```
2093  15   37:3    56        LINE:=LINE-1;
2094  15   37:3    64        GETLEADING; (* RESET LINESTART AND STUFFSTART *)
2095  15   37:2    67      END;
2096  15   37:1    69    CURSOR:=MAX(STUFFSTART,MAX(CURSOR-REPEATFACTOR,1));
2097  15   37:1    86    IF LINE<1 THEN CENTER;
2098  15   37:1    95    FINDXY(X,LINE);
2099  15   37:0   104 END (* LEFTMOVE *);
2100  15   37:0   118
2101  15   38:0     1 PROCEDURE RIGHTMOVE;
2102  15   38:0     1 VAR
2103  15   38:0     1   EOLPTR: PTRTYPE;
2104  15   38:0     0 BEGIN
2105  15   38:1     0   EOLPTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR;
2106  15   38:1    14   WHILE (EOLPTR<CURSOR+REPEATFACTOR) AND (EOLPTR<BUFCOUNT-1) DO
2107  15   38:2    27     BEGIN
2108  15   38:3    27       REPEATFACTOR:=REPEATFACTOR-(EOLPTR-CURSOR+1);
2109  15   38:3    36       CURSOR:=EOLPTR+1; (* BEGINNING OF THE LINE BELOW *)
2110  15   38:3    41       GETLEADING;
2111  15   38:3    44       CURSOR:=STUFFSTART;
2112  15   38:3    47       LINE:=LINE+1;
2113  15   38:3    55       IF LINE=SCREENHEIGHT+1 THEN SCROLLMARK:=LINESTART;
2114  15   38:3    68       EOLPTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR
2115  15   38:2    78     END;
2116  15   38:1    84   IF LINE>SCREENHEIGHT THEN
2117  15   38:2    91     IF (LINE-SCREENHEIGHT>=SCREENHEIGHT) OR (INDELETE) THEN
2118  15   38:3   104       CENTER
2119  15   38:2   104     ELSE
2120  15   38:3   108       SCROLLUP(SCROLLMARK,LINE-SCREENHEIGHT);
2121  15   38:1   118   CURSOR:=MIN(BUFCOUNT-1,CURSOR+REPEATFACTOR);
2122  15   38:1   131   FINDXY(X,LINE);
2123  15   38:0   140 END(* RIGHTMOVE *);
2124  15   38:0   154
2125  15   39:0     1 PROCEDURE LINEMOVE(REPEATFACTOR: INTEGER);
2126  15   39:0     2 VAR I: INTEGER;
2127  15   39:0     0 BEGIN
2128  15   39:1     0   I:=1;
2129  15   39:1     3   IF DIRECTION='<' THEN
2130  15   39:2     8     BEGIN
2131  15   39:3     8       WHILE (I<=REPEATFACTOR) AND (CURSOR>1) DO
2132  15   39:4    17         BEGIN
2133  15   39:5    17           IF EBUF^[CURSOR]=CHR(EOL) THEN CURSOR:=CURSOR-1; (* NULL LINE   SE *)
```

```
2134  15   39:5    29          CURSOR:=SCAN(-MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR; (* 1 UP *)
2135  15   39:5    44          IF CURSOR>=1 THEN BEGIN LINE:=LINE-1; I:=I+1 END;
2136  15   39:4    62        END;
2137  15   39:3    64      CURSOR:=MAX(1,CURSOR); (* BACK INTO REALITY *)
2138  15   39:3    73      ATEND:= (CURSOR=1);
2139  15   39:3    79      IF LINE<1 THEN CENTER
2140  15   39:2    86    END
2141  15   39:1    88  ELSE
2142  15   39:2    90    BEGIN (* DIRECTION='>' *)
2143  15   39:3    90      WHILE (I<=REPEATFACTOR) AND (CURSOR<BUFCOUNT-1) DO
2144  15   39:4   101        BEGIN
2145  15   39:5   101          CURSOR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+CURSOR+1; (*1 DOWN *)
2146  15   39:5   117          IF CURSOR<BUFCOUNT THEN
2147  15   39:6   122            BEGIN
2148  15   39:7   122              I:=I+1; LINE:=LINE+1;
2149  15   39:7   135              IF LINE=SCREENHEIGHT+1 THEN SCROLLMARK:=CURSOR;
2150  15   39:6   148            END
2151  15   39:4   148        END;
2152  15   39:3   150      ATEND:= (CURSOR>=BUFCOUNT-1);
2153  15   39:3   158      IF LINE>SCREENHEIGHT THEN
2154  15   39:4   165        IF (LINE-SCREENHEIGHT>=SCREENHEIGHT) OR
2155  15   39:4   172           INREPLACE OR (COMMAND=PARAC) OR INDELETE
2156  15   39:4   180         THEN
2157  15   39:5   186          CENTER
2158  15   39:4   186        ELSE
2159  15   39:5   190          SCROLLUP(SCROLLMARK,LINE-SCREENHEIGHT);
2160  15   39:3   200      CURSOR:=MIN(CURSOR,BUFCOUNT-1)
2161  15   39:2   204    END;
2162  15   39:1   211  GETLEADING;
2163  15   39:1   214  CURSOR:=STUFFSTART; (* FORCED TO BEGINNING OF STUFF *)
2164  15   39:1   217  X:=BLANKS;
2165  15   39:0   221 END(* LINEMOVE *);
2166  15   39:0   238
2167  15   40:0     1 PROCEDURE JUMPBEGIN;
2168  15   40:0     0 BEGIN
2169  15   40:1     0   CURSOR:=1; CENTERCURSOR(TRASH,1,FALSE)
2170  15   40:0     8 END;
2171  15   40:0    24
2172  15   41:0     1 PROCEDURE JUMPEND;
2173  15   41:0     0 BEGIN
2174  15   41:1     0   CURSOR:=BUFCOUNT-1; CENTERCURSOR(TRASH,SCREENHEIGHT,FALSE)
```

```
2175  15   41:0    10 END;
2176  15   41:0    26
2177  15   42:0     1 PROCEDURE ADJUSTING;
2178  15   42:0     1 LABEL 1;
2179  15   42:0     1 TYPE
2180  15   42:0     1   MODES=(RELATIVE,LEFTJ,RIGHTJ,CENTER);
2181  15   42:0     1 VAR
2182  15   42:0     1   LLENGTH,TDELTA,I: INTEGER;
2183  15   42:0     4   SAVEDIR: CHAR;
2184  15   42:0     5   MODE: MODES;
2185  15   42:0     6
2186  15   43:0     1 PROCEDURE DOIT(DELTA:INTEGER);
2187  15   43:0     2 VAR
2188  15   43:0     2   EOLDIST: INTEGER;
2189  15   43:0     3   T: PACKED ARRAY [0..MAXSTRING] OF CHAR;
2190  15   43:0     0 BEGIN
2191  15   43:1     0   GETLEADING; (* SET LINESTART, STUFFSTART, AND BLANKS *)
2192  15   43:1     3   IF BLANKS+DELTA<0 THEN DELTA:=-BLANKS;
2193  15   43:1    14   IF (EBUF^[LINESTART]=CHR(DLE)) AND (STUFFSTART-LINESTART=2) THEN
2194  15   43:2    27     X:=ORD(EBUF^[LINESTART+1])+DELTA-32
2195  15   43:1    34   ELSE
2196  15   43:2    41     BEGIN
2197  15   43:3    41       IF STUFFSTART-LINESTART>2 THEN
2198  15   43:4    48         MOVELEFT(EBUF^[STUFFSTART],EBUF^[LINESTART+2],BUFCOUNT-STUFFSTART)
2199  15   43:3    59       ELSE
2200  15   43:4    61         BEGIN
2201  15   43:5    61           IF BUFCOUNT>BUFSIZE-100 THEN
2202  15   43:6    68             BEGIN
2203  15   43:7    68               ERROR('BUFFER OVERFLOW',NONFATAL);
2204  15   43:7    90               EXIT(ADJUSTING)
2205  15   43:6    94             END
2206  15   43:5    94           ELSE
2207  15   43:6    96             MOVERIGHT(EBUF^[STUFFSTART],EBUF^[LINESTART+2],BUFCOUNT-STUFFSTART);
2208  15   43:4   107         END;
2209  15   43:3   107       IF LINESTART+2<>STUFFSTART THEN
2210  15   43:4   114         BEGIN
2211  15   43:5   114           READJUST(LINESTART,LINESTART+2-STUFFSTART);
2212  15   43:5   123           BUFCOUNT:=BUFCOUNT+LINESTART+2-STUFFSTART;
2213  15   43:4   132         END;
2214  15   43:3   132       EBUF^[LINESTART]:=CHR(DLE);
2215  15   43:3   136       X:=BLANKS+DELTA;
```

```
2216  15   43:2   142      END;
2217  15   43:1   142    EBUF^[LINESTART+1]:=CHR(X+32);
2218  15   43:1   152    CURSOR:=LINESTART+2; GETLEADING;
2219  15   43:1   160    GOTOXY(0,LINE); ERASETOEOL(0,LINE); (* ERASE THE LINE *)
2220  15   43:1   174    LINEOUT(LINESTART,BYTES,BLANKS,LINE); GOTOXY(X,LINE);
2221  15   43:0   193  END(* DOIT *);
2222  15   43:0   206
2223  15   42:0     0  BEGIN (* ADJUSTING *)
2224  15   42:1     0    WITH PAGEZERO DO
2225  15   42:2     0      BEGIN
2226  15   42:3     0        SAVEDIR:=DIRECTION; EXITPROMPT:=FALSE; INDELETE:=FALSE; LASTPAT:=CURSOR;
2227  15   42:3    14        INREPLACE:=TRUE;
2228  15   42:3    18        PROMPTLINE:=
2229  15   42:3    21    ' ADJUST: L(JUST R(JUST C(ENTER <LEFT,RIGHT,UP,DOWN-ARROWS> [<ETX> TO LEAVE]';
2230  15   42:3   101        PROMPT; NEEDPROMPT:=TRUE;
2231  15   42:3   108        MODE:=RELATIVE;
2232  15   42:3   111        SHOWCURSOR;
2233  15   42:3   114        FINDXY(X,LINE);
2234  15   42:3   123        TDELTA:=0;
2235  15   42:3   126        REPEAT
2236  15   42:4   126          CH:=GETCH;
2237  15   42:4   133          COMMAND:=MAPTOCOMMAND(CH);
2238  15   42:4   141          INFINITY:=FALSE;
2239  15   42:4   145          IF COMMAND=SLASHC THEN
2240  15   42:5   150            BEGIN
2241  15   42:6   150              REPEATFACTOR:=1; INFINITY:=TRUE; CH:=GETCH; COMMAND:=TRANSLATE[CH]
2242  15   42:5   170            END
2243  15   42:4   173          ELSE
2244  15   42:5   175            IF COMMAND=DIGIT THEN REPEATFACTOR:=GETNUM ELSE REPEATFACTOR:=1;
2245  15   42:4   192          IF COMMAND IN [UP,DOWN] THEN
2246  15   42:5   204            BEGIN
2247  15   42:6   204              IF COMMAND=UP THEN DIRECTION:='<' ELSE DIRECTION:='>';
2248  15   42:6   217              I:=1;
2249  15   42:6   220              ATEND:=FALSE;
2250  15   42:6   224              WHILE NOT ATEND AND ((I<=REPEATFACTOR) OR INFINITY) DO
2251  15   42:7   238                BEGIN
2252  15   42:8   238                  I:=I+1;
2253  15   42:8   243                  LINEMOVE(1);
2254  15   42:8   246                  IF NOT ATEND THEN
2255  15   42:9   252                    BEGIN
2256  15   42:0   252                      IF MODE=RELATIVE THEN DOIT(TDELTA)
```

```
2257  15    42:0    258                ELSE
2258  15    42:1    262                  BEGIN
2259  15    42:2    262                    LLENGTH:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[STUFFSTART]);
2260  15    42:2    274                    CASE MODE OF
2261  15    42:2    277                      LEFTJ:  DOIT(LMARGIN-BLANKS);
2262  15    42:2    286                      RIGHTJ: DOIT((RMARGIN-LLENGTH+1)-BLANKS);
2263  15    42:2    299                      CENTER:
2264  15    42:3    299                      DOIT(((RMARGIN-LMARGIN+1)-LLENGTH) DIV 2-BLANKS+LMARGIN)
2265  15    42:2    318                    END (* CASE *)
2266  15    42:1    336                  END (* ELSE *)
2267  15    42:9    336              END; (* IF NOT ATEND *)
2268  15    42:7    336          END (* WHILE ... *)
2269  15    42:5    336      END
2270  15    42:4    338  ELSE
2271  15    42:5    340    IF COMMAND=LEFT THEN
2272  15    42:6    345      BEGIN
2273  15    42:7    345        DOIT(-REPEATFACTOR); TDELTA:=TDELTA-REPEATFACTOR; MODE:=RELATIVE
2274  15    42:6    354      END
2275  15    42:5    357    ELSE
2276  15    42:6    359      IF COMMAND=RIGHT THEN
2277  15    42:7    364        BEGIN
2278  15    42:8    364          DOIT(REPEATFACTOR); TDELTA:=TDELTA+REPEATFACTOR; MODE:=RELATIVE
2279  15    42:7    372        END
2280  15    42:6    375      ELSE
2281  15    42:7    377        IF COMMAND IN [LISTC,REPLACEC,COPYC] THEN
2282  15    42:8    385          BEGIN
2283  15    42:9    385            GETLEADING;
2284  15    42:9    388            LLENGTH:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[STUFFSTART]);
2285  15    42:9    400            IF COMMAND=LISTC THEN
2286  15    42:0    405              BEGIN MODE:=LEFTJ; DOIT(LMARGIN-BLANKS) END
2287  15    42:9    415            ELSE
2288  15    42:0    417              IF COMMAND=REPLACEC THEN
2289  15    42:1    422                BEGIN MODE:=RIGHTJ; DOIT((RMARGIN-LLENGTH+1)-BLANKS) END
2290  15    42:0    436              ELSE (* COMMAND=COPYC *)
2291  15    42:1    438                BEGIN
2292  15    42:2    438                  MODE:=CENTER;
2293  15    42:2    441                  DOIT(((RMARGIN-LMARGIN+1)-LLENGTH) DIV 2-BLANKS+LMARGIN)
2294  15    42:1    460                END
2295  15    42:8    462          END
2296  15    42:7    462        ELSE
2297  15    42:8    464      IF CH<>CHR(ETX) T  BEGIN ERRWAIT; SHOWCURSOR END;
```

```
2298  15    42:4    477        1: UNTIL CH=CHR(ETX);
2299  15    42:3    484        DIRECTION:=SAVEDIR;
2300  15    42:2    487      END;
2301  15    42:0    487 END;
2302  15    42:0    508
2303  15    44:0      1 PROCEDURE TABBY;
2304  15    44:0      1 (* SCAN ALONG THE LINE UNTIL YOU EITHER HIT A TABSTOP OR THE END OF THE LINE *)
2305  15    44:0      1 VAR
2306  15    44:0      1   NEWX,ENDX,I,NUMTODO: INTEGER;
2307  15    44:0      0 BEGIN
2308  15    44:1      0   NUMTODO:=REPEATFACTOR;
2309  15    44:1      3   FOR I:=1 TO NUMTODO DO
2310  15    44:2     14   BEGIN
2311  15    44:3     14     REPEATFACTOR:=1;
2312  15    44:3     17     IF DIRECTION='>' THEN RIGHTMOVE ELSE LEFTMOVE;
2313  15    44:3     28     NEWX:=X;
2314  15    44:3     33     WITH PAGEZERO DO
2315  15    44:4     33       BEGIN
2316  15    44:5     33         IF DIRECTION='>' THEN
2317  15    44:6     38           BEGIN
2318  15    44:7     38             ENDX:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[CURSOR])+X;
2319  15    44:7     54             WHILE (TABSTOP[NEWX]=NONE) AND (NEWX<ENDX) DO NEWX:=NEWX+1;
2320  15    44:6     77           END
2321  15    44:5     77         ELSE
2322  15    44:6     79           BEGIN
2323  15    44:7     79             GETLEADING;
2324  15    44:7     82             WHILE (TABSTOP[NEWX]=NONE) AND (NEWX>BLANKS) DO NEWX:=NEWX-1;
2325  15    44:6    105           END;
2326  15    44:5    105         REPEATFACTOR:=ABS(NEWX-X);
2327  15    44:5    113         IF DIRECTION='>' THEN RIGHTMOVE ELSE LEFTMOVE;
2328  15    44:4    124       END; (* WITH *)
2329  15    44:2    124   END (* FOR *)
2330  15    44:0    124 END;
2331  15    44:0    150
2332  15    45:0      1 PROCEDURE MOVING;
2333  15    45:0      1 VAR
2334  15    45:0      1   SAVEX: INTEGER;
2335  15    45:0      0 BEGIN
2336  15    45:1      0   INDELETE:=FALSE;
2337  15    45:1      4   INREPLACE:=FALSE;
2338  15    45:1      8   EXITPROMPT:=FALSE;
```

```
2339  15    45:1     12   IF INFINITY THEN
2340  15    45:2     17     BEGIN
2341  15    45:3     17       CASE COMMAND OF
2342  15    45:3     20         UP,LEFT: JUMPBEGIN;
2343  15    45:3     24         DOWN,RIGHT: JUMPEND;
2344  15    45:3     28         PARAC,SPACE,ADVANCE,TAB: IF DIRECTION='<' THEN JUMPBEGIN ELSE JUMPEND
2345  15    45:3     37       END;
2346  15    45:3     84       NEEDPROMPT:=TRUE;
2347  15    45:3     88       NEXTCOMMAND;
2348  15    45:3     90       EXIT(MOVEIT)
2349  15    45:2     94     END;
2350  15    45:1     94   FINDXY(X,LINE);
2351  15    45:1    103   REPEAT
2352  15    45:2    103     OLDX:=X; OLDLINE:=LINE;
2353  15    45:2    115     CASE COMMAND OF
2354  15    45:2    118       LEFT: LEFTMOVE;
2355  15    45:2    122       RIGHT: RIGHTMOVE;
2356  15    45:2    126       SPACE: IF DIRECTION='<' THEN LEFTMOVE ELSE RIGHTMOVE;
2357  15    45:2    139       UP: UPMOVE;
2358  15    45:2    143       DOWN: DOWNMOVE;
2359  15    45:2    147       ADVANCE: LINEMOVE(REPEATFACTOR);
2360  15    45:2    152       PARAC:
2361  15    45:3    152         IF REPEATFACTOR>1000 THEN ERROR('TOO MANY',NONFATAL)
2362  15    45:3    171         ELSE LINEMOVE(SCREENHEIGHT*REPEATFACTOR);
2363  15    45:2    183       TAB: TABBY
2364  15    45:2    183     END;
2365  15    45:2    230     IF EXITPROMPT OR (COMMAND=PARAC) THEN
2366  15    45:3    239       GOTOXY(X,LINE)
2367  15    45:2    248     ELSE
2368  15    45:3    250       IF LINE=OLDLINE THEN
2369  15    45:4    259         BEGIN
2370  15    45:5    259           IF X=OLDX+1 THEN CONTROL(FS) ELSE IF X=OLDX-1 THEN WRITE(CHR(BSPCE))
2371  15    45:6    295           ELSE GOTOXY(X,LINE)
2372  15    45:4    306         END
2373  15    45:3    306       ELSE
2374  15    45:4    308         IF X=OLDX THEN
2375  15    45:5    317           BEGIN
2376  15    45:6    317             IF LINE=OLDLINE+1 THEN WRITE(CHR(LF))
2377  15    45:6    336             ELSE IF LINE=OLDLINE-1 THEN CONTROL(US)
2378  15    45:7    350             ELSE GOTOXY(X,LINE);
2379  15    45:5    364           END
```

```
2380  15   45:4   364          ELSE
2381  15   45:5   366            GOTOXY(X,LINE);
2382  15   45:2   375      REPEATFACTOR:=1;
2383  15   45:2   378      NEXTCOMMAND
2384  15   45:1   378    UNTIL NOT (COMMAND IN [UP,DOWN,LEFT,RIGHT,ADVANCE,SPACE,TAB]);
2385  15   45:1   393    IF EXITPROMPT THEN PROMPT;
2386  15   45:1   401    SHOWCURSOR;
2387  15   45:0   404 END (* MOVING *);
2388  15   45:0   418
2389  15   46:0     1 PROCEDURE PUTITBACK(C1,C2: PTRTYPE);
2390  15   46:0     3 VAR
2391  15   46:0     3   PTR: PTRTYPE;
2392  15   46:0     4   INDENT,LOFF: INTEGER;
2393  15   46:0     0 BEGIN
2394  15   46:1     0   PTR:=C1;
2395  15   46:1     3   WHILE PTR<=C2 DO
2396  15   46:2     8     BEGIN
2397  15   46:3     8       IF EBUF^[PTR]=CHR(EOL) THEN
2398  15   46:4    15         BEGIN
2399  15   46:5    15           PTR:=PTR+1; WRITELN;
2400  15   46:5    26           INDENT:=LEADBLANKS(PTR,LOFF);
2401  15   46:5    36           IF (PTR<C2) AND (INDENT>0) THEN
2402  15   46:6    45             WRITE(' ':INDENT);
2403  15   46:5    53           PTR:=PTR+LOFF
2404  15   46:4    54         END
2405  15   46:3    58       ELSE
2406  15   46:4    60         BEGIN WRITE(EBUF^[PTR]); PTR:=PTR+1 END;
2407  15   46:2    75     END;
2408  15   46:0    77 END;
2409  15   46:0    92
2410  15   33:0     1 PROCEDURE CLEAR(*X1,Y1,X2,Y2: INTEGER*);
2411  15   33:0     5 (* SCREEN CO-ORDINATE (X1,Y1) IS ASSUMED TO BE BEFORE (X2,Y2).  THIS
2412  15   33:0     5    PROCEDURE TAKES THESE CO-ORDINATES AND CLEARS (WRITES BLANKS) OVER
2413  15   33:0     5    THE SCREEN BETWEEN THEM (INCLUSIVE)  *)
2414  15   33:0     5 VAR XX,I: INTEGER;
2415  15   33:0     0 BEGIN
2416  15   33:1     0   GOTOXY(X1,Y1);
2417  15   33:1     5   XX:=X1;
2418  15   33:1     8   FOR I:=Y1 TO Y2-1 DO BEGIN IF I<>0 THEN ERASETOEOL(XX,I); XX:=0; WRITELN END;
2419  15   33:1    47   IF Y1<>Y2 THEN FOR I:=0 TO X2 DO WRITE(' ')
2420  15   33:1    71   ELSE FOR I:=X1 TO X2 DO WRITE(' ')
```

```
2421  15   33:0    99 END;
2422  15   33:0   124
2423  15   47:0     1 PROCEDURE RESOLVESCREEN;
2424  15   47:0     1 VAR
2425  15   47:0     1   X1,X2,Y1,Y2,SAVE: INTEGER;
2426  15   47:0     6   C1,C2: PTRTYPE;
2427  15   47:0     0 BEGIN
2428  15   47:1     0   X1:=X;     Y1:=LINE;
2429  15   47:1    10   X2:=OLDX; Y2:=OLDLINE;
2430  15   47:1    20   IF NEWDIST>DIST THEN
2431  15   47:2    29     BEGIN C1:=CURSOR-1; C2:=OLDCURSOR; X1:=X1-1 END
2432  15   47:1    44   ELSE
2433  15   47:2    46     IF NEWDIST<DIST THEN
2434  15   47:3    55       BEGIN C2:=OLDCURSOR-1; C1:=CURSOR; X2:=X2-1 END
2435  15   47:2    70     ELSE
2436  15   47:3    72       EXIT(RESOLVESCREEN);
2437  15   47:1    76   IF (Y1>Y2) OR ((Y1=Y2) AND (X1>X2)) THEN
2438  15   47:2    89     BEGIN
2439  15   47:3    89       SAVE:=C1; C1:=C2; C2:=SAVE;
2440  15   47:3    98       SAVE:=Y1; Y1:=Y2; Y2:=SAVE;
2441  15   47:3   107       SAVE:=X1; X1:=X2; X2:=SAVE
2442  15   47:2   113     END;
2443  15   47:1   116   IF ABS(NEWDIST)>ABS(DIST) THEN
2444  15   47:2   127     CLEAR(X1,Y1,X2,Y2)
2445  15   47:1   131   ELSE
2446  15   47:2   135     BEGIN
2447  15   47:3   135       GOTOXY(X1,Y1);
2448  15   47:3   140       PUTITBACK(C1,C2)
2449  15   47:2   142     END;
2450  15   47:1   144   GOTOXY(X,LINE)
2451  15   47:0   153 END;
2452  15   47:0   166
2453  15   48:0     1 PROCEDURE DELETING;
2454  15   48:0     1 LABEL 1;
2455  15   48:0     1 VAR
2456  15   48:0     1   ATBOL,ANCHOR,SAVE: PTRTYPE;
2457  15   48:0     4   OK,ATBOT,NOMOVE: BOOLEAN;
2458  15   48:0     7   STARTLINE: INTEGER;
2459  15   48:0     8
2460  15   48:0     0 BEGIN
2461  15   48:1     0   DOFFSCREEN:=FALSE; INDELETE:=TRUE; INREPLACE:=FALSE; EXITPROMPT:=FALSE
```

```
2462  15   48:1    16    ANCHOR:=CURSOR; NEWDIST:=0;
2463  15   48:1    23    GETLEADING; ATBOL:=LINESTART; ATBOT:=(CURSOR=STUFFSTART);
2464  15   48:1    34    PROMPTLINE:=
2465  15   48:1    37  * DELETE: < > <MOVING COMMANDS> [<ETX> TO DELETE, <ESC> TO ABORT]';
2466  15   48:1   106    PROMPT; NEEDPROMPT:=TRUE;
2467  15   48:1   113    SHOWCURSOR;
2468  15   48:1   116    FINDXY(X,LINE);
2469  15   48:1   125    STARTLINE:=LINE;
2470  15   48:1   130    REPEAT
2471  15   48:2   130      OLDCURSOR:=CURSOR;
2472  15   48:2   134      DIST:=NEWDIST;
2473  15   48:2   140      OLDX:=X; OLDLINE:=LINE;
2474  15   48:2   152      CH:=GETCH;
2475  15   48:2   159      COMMAND:=TRANSLATE[CH];
2476  15   48:2   168      IF COMMAND=DIGIT THEN REPEATFACTOR:=GETNUM ELSE REPEATFACTOR:=1;
2477  15   48:2   185      IF COMMAND IN [REVERSEC..DIGIT,ADVANCE,SPACE] THEN
2478  15   48:3   196        BEGIN
2479  15   48:4   196          CASE COMMAND OF
2480  15   48:4   199            LEFT: LEFTMOVE;
2481  15   48:4   203            RIGHT: RIGHTMOVE;
2482  15   48:4   207            SPACE: IF DIRECTION='<' THEN LEFTMOVE ELSE RIGHTMOVE;
2483  15   48:4   220            UP: UPMOVE;
2484  15   48:4   224            DOWN: DOWNMOVE;
2485  15   48:4   228            ADVANCE: LINEMOVE(REPEATFACTOR);
2486  15   48:4   233            REVERSEC,FORWARDC:
2487  15   48:5   233              BEGIN
2488  15   48:6   233                IF COMMAND=REVERSEC THEN
2489  15   48:7   238                  DIRECTION:='<'
2490  15   48:6   238                ELSE
2491  15   48:7   243                  DIRECTION:='>';
2492  15   48:6   246                GOTOXY(0,0); WRITE(DIRECTION); GOTOXY(X,LINE)
2493  15   48:5   268              END;
2494  15   48:4   270            TAB: TABBY
2495  15   48:4   270          END;
2496  15   48:4   304          NEWDIST:=CURSOR-ANCHOR;
2497  15   48:4   310          RESOLVESCREEN;
2498  15   48:3   312        END
2499  15   48:2   312      ELSE
2500  15   48:3   314        IF (CH<>CHR(ESC)) AND (CH<>CHR(ETX)) THEN
2501  15   48:4   327          BEGIN ERRWAIT; GOTOXY(X,LINE) END
2502  15   48:1   339    UNTIL (CH IN [CHR(ETX),CHR(ESC)]);
```

```
2503  15   48:1   352   IF CH=CHR(ETX) THEN
2504  15   48:2   359     BEGIN
2505  15   48:3   359       GETLEADING; (* INDENTATION FIXUP *)
2506  15   48:3   362       IF ATBOT AND (CURSOR=STUFFSTART) THEN
2507  15   48:4   369         BEGIN CURSOR:=LINESTART; SAVE:=ANCHOR; ANCHOR:=ATBOL END;
2508  15   48:3   378       IF OKTODEL(CURSOR,ANCHOR) THEN
2509  15   48:4   387         BEGIN
2510  15   48:5   387           READJUST(MIN(CURSOR,ANCHOR),-ABS(CURSOR-ANCHOR));
2511  15   48:5   402           COPYLINE:=(CURSOR=LINESTART) AND ATBOT;
2512  15   48:5   410           IF ANCHOR<CURSOR THEN
2513  15   48:6   415             MOVELEFT(EBUF^[CURSOR],EBUF^[ANCHOR],BUFCOUNT-CURSOR)
2514  15   48:5   424           ELSE
2515  15   48:6   426             MOVELEFT(EBUF^[ANCHOR],EBUF^[CURSOR],BUFCOUNT-ANCHOR);
2516  15   48:5   435           BUFCOUNT:=BUFCOUNT-ABS(CURSOR-ANCHOR);
2517  15   48:5   443           CURSOR:=MIN(CURSOR,ANCHOR);
2518  15   48:5   452           GETLEADING; CURSOR:=MAX(STUFFSTART,CURSOR)
2519  15   48:4   457         END
2520  15   48:3   464       ELSE
2521  15   48:4   466         CURSOR:=SAVE
2522  15   48:2   466     END
2523  15   48:1   469   ELSE
2524  15   48:2   471     BEGIN
2525  15   48:3   471       COPYLINE:=FALSE; COPYOK:=TRUE;
2526  15   48:3   479       COPYSTART:=MIN(CURSOR,ANCHOR);
2527  15   48:3   489       COPYLENGTH:=ABS(CURSOR-ANCHOR);
2528  15   48:3   496       CURSOR:=ANCHOR;
2529  15   48:2   499     END;
2530  15   48:1   499   1:INDELETE:=FALSE;
2531  15   48:1   503   OK:=(LINE=STARTLINE) AND NOT DOFFSCREEN;
2532  15   48:1   515   UPSCREEN(OK,NOT OK,LINE);
2533  15   48:1   524   NEXTCOMMAND;
2534  15   48:0   526 END;
2535  15   48:0   540
2536  15   31:0     0 BEGIN
2537  15   31:1     0   IF COMMAND=DELETEC THEN
2538  15   31:2     5     DELETING
2539  15   31:1     5   ELSE
2540  15   31:2     9     IF COMMAND=ADJUSTC THEN
2541  15   31:3    14       BEGIN ADJUSTING; NEXTCOMMAND END
2542  15   31:2    18     ELSE MOVING;
2543  15   31:0    22 END;
```

```
2544  15   31:0     34
2545  15   31:0     34
2546  15   31:0     34 (*$IF I N D   &   R E P L A C E*)
2547  15   31:0     34
2548  15    8:0      1 PROCEDURE FIND;
2549  15    8:0      1 LABEL 1;
2550  15    8:0      1 VAR
2551  15    8:0      1   ALREADYSAIDGO,THERE,FOUND,LASTPATTERN: BOOLEAN;
2552  15    8:0      5   TRASH,COULDBE,PLENGTH,START,STOP,NEXTSTART: INTEGER;
2553  15    8:0     11   NEXT,PTR: PTRTYPE;
2554  15    8:0     13   MODE: (LITERAL,TOKEN);
2555  15    8:0     14   I: INTEGER;
2556  15    8:0     15   DELIMITER: CHAR;
2557  15    8:0     16   JUSTIN: BOOLEAN;
2558  15    8:0     17   POSSIBLE,PAT: PTYPE;
2559  15    8:0    145   USEOLD,VERIFY: BOOLEAN;
2560  15    8:0    147
2561  15   49:0      1 PROCEDURE NEXTCH;
2562  15   49:0      0 BEGIN
2563  15   49:1      0   CH:=GETCH;
2564  15   49:1      7   IF CH=CHR(ESC) THEN
2565  15   49:2     14       BEGIN
2566  15   49:3     14         IF NOT JUSTIN THEN REDISPLAY;
2567  15   49:3     23         SHOWCURSOR; NEXTCOMMAND;
2568  15   49:3     28         EXIT(FIND);
2569  15   49:2     32       END;
2570  15   49:1     32   IF (CH=CHR(EOL)) AND JUSTIN THEN
2571  15   49:2     41     BEGIN
2572  15   49:3     41       JUSTIN:=FALSE;
2573  15   49:3     45       BLANKCRT(1)
2574  15   49:2     46     END
2575  15   49:1     49   ELSE
2576  15   49:2     51     WRITE(CH);
2577  15   49:0     59 END;
2578  15   49:0     72
2579  15   50:0      1 PROCEDURE SKIP;
2580  15   50:0      0 BEGIN
2581  15   50:1      0   WHILE CH IN [CHR(SP),CHR(HT),CHR(EOL)] DO NEXTCH
2582  15   50:0     12 END;
2583  15   50:0     30
2584  15   51:0      1 PROCEDURE OPTIONS;
```

```
2585  15   51:0     0 BEGIN
2586  15   51:1     0   REPEAT
2587  15   51:2     0     CH:=UCLC(CH);
2588  15   51:2     8     IF CH='L' THEN
2589  15   51:3    13       BEGIN MODE:=LITERAL; NEXTCH END
2590  15   51:2    19     ELSE
2591  15   51:3    21       IF CH='V' THEN
2592  15   51:4    26         BEGIN VERIFY:=TRUE; NEXTCH END
2593  15   51:3    33       ELSE
2594  15   51:4    35         IF CH='T' THEN
2595  15   51:5    40           BEGIN MODE:=TOKEN; NEXTCH END;
2596  15   51:2    46     CH:=UCLC(CH);
2597  15   51:1    54   UNTIL NOT ((CH='V') OR (CH='T') OR (CH='L'));
2598  15   51:1    68   SKIP;
2599  15   51:1    70   IF (CH='S') OR (CH='S') THEN USEOLD:=TRUE;
2600  15   51:0    84 END;
2601  15   51:0    98
2602  15   52:D     1 PROCEDURE PARSESTRING(VAR PATTERN: PTYPE; VAR PLENGTH: INTEGER);
2603  15   52:D     3 VAR I,J: INTEGER;
2604  15   52:0     0 BEGIN
2605  15   52:1     0   SKIP;
2606  15   52:1     2   IF CH IN C'A'..'Z','A'..'Z','0'..'9',CHR(BS)] THEN
2607  15   52:2    31     BEGIN
2608  15   52:3    31       ERROR('INVALID DELIMITER.',NONFATAL);
2609  15   52:3    56       IF NOT JUSTIN THEN REDISPLAY;
2610  15   52:3    65       NEXTCOMMAND;
2611  15   52:3    67       EXIT(FIND);
2612  15   52:2    71     END;
2613  15   52:1    71   DELIMITER:=CH;
2614  15   52:1    75   I:=0;
2615  15   52:1    78   REPEAT
2616  15   52:2    78     NEXTCH;
2617  15   52:2    80     IF CH=CHR(BS) THEN
2618  15   52:3    87       BEGIN
2619  15   52:4    87         IF (PATTERNCI]<>CHR(EOL)) AND (I>0) THEN (* DON'T GO OVERBOARD! *)
2620  15   52:5    98           BEGIN
2621  15   52:6    98             WRITE(' ',CHR(BS));
2622  15   52:6   116             I:=I-1
2623  15   52:5   117           END
2624  15   52:4   121         ELSE CONTROL(FS); (* MAKE UP FOR THE <BS> NEXTCH WROTE OUT *)
2625  15   52:3   127       END
```

```
2626  15   52:2   127      ELSE
2627  15   52:3   129        BEGIN
2628  15   52:4   129          PATTERN[I]:=CH;
2629  15   52:4   133          I:=I+1
2630  15   52:3   134        END;
2631  15   52:1   138    UNTIL (CH=DELIMITER) OR (I>=MAXSTRING);
2632  15   52:1   149    IF I>=MAXCHAR THEN
2633  15   52:2   156      BEGIN
2634  15   52:3   156        ERROR('YOUR PATTERN IS TOO LONG',NONFATAL);
2635  15   52:3   187        IF NOT JUSTIN THEN REDISPLAY;
2636  15   52:3   196        NEXTCOMMAND: EXIT(FIND)
2637  15   52:2   202      END;
2638  15   52:1   202    PLENGTH:=I-1;
2639  15   52:0   207  END (* PARSESTRING *);
2640  15   52:0   222
2641  15   53:D     3  FUNCTION OK(PTR: PTRTYPE): BOOLEAN;
2642  15   53:D     4  (* COMPARE PAT AGAINST THE BUFFER *)
2643  15   53:D     4  VAR I: INTEGER;
2644  15   53:0     0  BEGIN
2645  15   53:1     0    I:=0;
2646  15   53:1     3    WHILE (I<PLENGTH) AND (EBUF^[PTR+I]=PAT[I]) DO I:=I+1;
2647  15   53:1    29    OK:= I=PLENGTH;
2648  15   53:0    36  END;
2649  15   53:0    50
2650  15   54:D     1  PROCEDURE SKIPKIND3(VAR CURSOR: PTRTYPE);
2651  15   54:0     0  BEGIN
2652  15   54:0     0    (* SKIP OVER KIND3 CHARACTERS IN THE EBUF.  UPDATE THE CURSOR
2653  15   54:0     0       TO THE FIRST NON-KIND3 CHARACTER                          *)
2654  15   54:1     0    WHILE EBUF^[CURSOR] IN [CHR(SP),CHR(HT),CHR(DLE),CHR(EOL)] DO
2655  15   54:2    18      IF EBUF^[CURSOR]=CHR(DLE) THEN CURSOR:=CURSOR+2
2656  15   54:2    29      ELSE CURSOR:=CURSOR+1;
2657  15   54:0    42  END;
2658  15   54:0    56
2659  15   55:D     1  PROCEDURE SCANBACKWARD;
2660  15   55:D     1  LABEL 1;
2661  15   55:D     1  VAR
2662  15   55:D     1    LOC: PTRTYPE;
2663  15   55:D     2    CHTHERE: BOOLEAN;
2664  15   55:0     0  BEGIN
2665  15   55:1     0    CHTHERE:=TRUE;
2666  15   55:1     3    THERE:=FALSE;
```

```
2667  15   55:1     7   FILLCHAR(PAT[0],SIZEOF(PAT),' ');
2668  15   55:1    17   MOVELEFT(TARGET[START],PAT[0],PLENGTH);
2669  15   55:1    32   WHILE CHTHERE AND NOT THERE DO
2670  15   55:2    40     BEGIN
2671  15   55:3    40       1: IF PTR>=PLENGTH THEN (* POSSIBLY THERE *)
2672  15   55:4    49         LOC:=SCAN(-PTR,=PAT[0],EBUF^[PTR])
2673  15   55:3    66       ELSE
2674  15   55:4    70         LOC:=-PTR;
2675  15   55:3    76       IF LOC=-PTR THEN (* NOT THERE! *)
2676  15   55:4    84         BEGIN
2677  15   55:5    84           CHTHERE:=FALSE; THERE:=FALSE
2678  15   55:4    87         END
2679  15   55:3    91       ELSE
2680  15   55:4    93         BEGIN
2681  15   55:5    93           PTR:=PTR+LOC; NEXT:=PTR-1;
2682  15   55:5   109           IF EBUF^[PTR-1]=CHR(DLE) THEN BEGIN PTR:=NEXT; GOTO 1 END;
2683  15   55:5   128           IF OK(PTR) THEN THERE:=TRUE ELSE PTR:=NEXT
2684  15   55:4   143         END
2685  15   55:2   149     END;
2686  15   55:0   151 END;
2687  15   55:0   168
2688  15   56:D     1 PROCEDURE SCANFORWARD;
2689  15   56:D     1 LABEL 1;
2690  15   56:D     1 VAR
2691  15   56:D     1   MAXSCAN,LOC: INTEGER;
2692  15   56:D     3   CHTHERE: BOOLEAN;
2693  15   56:0     0 BEGIN
2694  15   56:1     0   CHTHERE:=TRUE;
2695  15   56:1     3   THERE:=FALSE;
2696  15   56:1     7   FILLCHAR(PAT[0],SIZEOF(PAT),' ');
2697  15   56:1    17   MOVELEFT(TARGET[START],PAT[0],PLENGTH);
2698  15   56:1    32   WHILE CHTHERE AND NOT THERE DO
2699  15   56:2    40    BEGIN
2700  15   56:3    40     1: MAXSCAN:=(BUFCOUNT-PLENGTH)-PTR+1;
2701  15   56:3    53     IF MAXSCAN>0 THEN (* STILL STUFF TO SCAN *)
2702  15   56:4    58       LOC:=SCAN(MAXSCAN,=PAT[0],EBUF^[PTR])
2703  15   56:3    72     ELSE
2704  15   56:4    76       LOC:=MAXSCAN; (* DUMMY UP 'NOT FOUND' CONDITION *)
2705  15   56:3    79     IF LOC=MAXSCAN THEN
2706  15   56:4    84       BEGIN CHTHERE:=FALSE; THERE:=FALSE END
2707  15   56:3    91     ELSE
```

```
2708  15   56:4    93        BEGIN
2709  15   56:5    93          PTR:=LOC+PTR; NEXT:=PTR+1;
2710  15   56:5   109          IF EBUF^[PTR-1]=CHR(DLE) THEN BEGIN PTR:=NEXT; GOTO 1 END;
2711  15   56:5   128          IF OK(PTR) THEN THERE:=TRUE ELSE PTR:=NEXT
2712  15   56:4   143        END
2713  15   56:2   149      END;
2714  15   56:0   151  END;
2715  15   56:0   168
2716  15   57:D     1  PROCEDURE GOFORIT;
2717  15   57:D     1
2718  15   58:D     1  PROCEDURE NEXTLINE;
2719  15   58:D     1  (* GIVEN NEXTSTART, CALCULATE THE START AND STOP FOR THE NEXT LINE *)
2720  15   58:0     0  BEGIN
2721  15   58:1     0    LASTPATTERN:=FALSE;
2722  15   58:1     4    START:=NEXTSTART;
2723  15   58:1    10    STOP:=MIN(TLENGTH-1,START+SCAN(TLENGTH-START,=CHR(EOL),TARGET[START]));
2724  15   58:1    45    IF STOP=TLENGTH-1 THEN BEGIN STOP:=MAX(STOP,0); LASTPATTERN:=TRUE END;
2725  15   58:1    72    NEXTSTART:=STOP+1;
2726  15   58:0    80  END;
2727  15   58:0    92
2728  15   59:D     1  PROCEDURE NEXTTOKEN;
2729  15   59:D     1  (* GIVEN NEXTSTART, CALCULATE START AND STOP *)
2730  15   59:0     0  BEGIN
2731  15   59:1     0    LASTPATTERN:=FALSE;
2732  15   59:1     4    START:=NEXTSTART;
2733  15   59:1    10    (* SKIP OVER LEADING KIND3 CHARACTERS *)
2734  15   59:1    10    WHILE (TARGET[START] IN [CHR(SP),CHR(EOL),CHR(HT)]) AND (START<TLENGTH-1) DO
2735  15   59:2    38      START:=START+1;
2736  15   59:1    48    STOP:=START;
2737  15   59:1    54    (* GET THE NEXT TOKEN *)
2738  15   59:1    54    WHILE (KIND[TARGET[START]]=KIND[TARGET[STOP+1]]) AND (STOP<TLENGTH-1) DO
2739  15   59:2    93      STOP:=STOP+1;
2740  15   59:1   103    STOP:=MIN(STOP,TLENGTH-1);
2741  15   59:1   119    (* TO ACCURATELY TEST FOR THE LAST TOKEN, SCAN OFF THE TRAILING KIND3
2742  15   59:1   119       CHARACTERS *)
2743  15   59:1   119    NEXTSTART:=STOP+1;
2744  15   59:1   127    WHILE (TARGET[NEXTSTART] IN [CHR(EOL),CHR(SP),CHR(HT)]) AND
2745  15   59:1   143          (NEXTSTART<TLENGTH) DO NEXTSTART:=NEXTSTART+1;
2746  15   59:1   163    IF NEXTSTART=TLENGTH THEN BEGIN STOP:=MAX(STOP,0); LASTPATTERN:=TRUE END;
2747  15   59:0   188  END;
2748  15   59:0   206
```

```
2749  15    57:0      0 BEGIN(* GOFORIT *)
2750  15    57:1      0   FOUND:=FALSE;
2751  15    57:1      4   NEXT:=PTR;
2752  15    57:1     10   REPEAT
2753  15    57:2     10     PTR:=NEXT; (* SET TO NEXT PLACE TO SCAN FOR *)
2754  15    57:2     16     NEXTSTART:=0;  (* FOOL NEXTLINE INTO GIVING US START AND STOP FOR LINE 1 *)
2755  15    57:2     20     IF MODE=LITERAL THEN NEXTLINE ELSE NEXTTOKEN;
2756  15    57:2     33     PLENGTH:=STOP-START+1;
2757  15    57:2     45     IF DIRECTION='>' THEN SCANFORWARD ELSE SCANBACKWARD;
2758  15    57:2     56     IF THERE THEN
2759  15    57:3     61       BEGIN
2760  15    57:4     61         COULDBE:=PTR;
2761  15    57:4     67         FOUND:=TRUE;
2762  15    57:4     71         WHILE (NOT LASTPATTERN) AND FOUND DO
2763  15    57:5     81           BEGIN
2764  15    57:6     81             IF MODE=LITERAL THEN NEXTLINE ELSE NEXTTOKEN;
2765  15    57:6     94             PTR:=PTR+PLENGTH;
2766  15    57:6    104             SKIPKIND3(PTR); (* GO PAST THE JUNK ON THE NEXT LINE *)
2767  15    57:6    109             PLENGTH:=STOP-START+1; (* FOR THE NEW LINE *)
2768  15    57:6    121             FILLCHAR(PAT[0],SIZEOF(PAT),' ');
2769  15    57:6    131             MOVELEFT(TARGET[START],PAT[0],PLENGTH);
2770  15    57:6    146             IF PTR+PLENGTH > BUFCOUNT THEN
2771  15    57:7    157               FOUND:=FALSE
2772  15    57:6    157             ELSE
2773  15    57:7    163               IF NOT OK(PTR) THEN FOUND:=FALSE;
2774  15    57:5    177           END;
2775  15    57:3    179       END;
2776  15    57:3    179     (* IN TOKEN MODE MAKE SURE THE FIRST AND LAST CHARACTERS
2777  15    57:3    179        OF THE TARGET ARE ON 'TOKEN BOUNDARIES' *)
2778  15    57:2    179     IF MODE=TOKEN THEN IF KIND[PAT[0]]=ORD('A') THEN IF FOUND THEN
2779  15    57:5    205       BEGIN
2780  15    57:6    205         IF ((COULDBE>2) AND (EBUF^[COULDBE-2]<>CHR(DLE))) OR
2781  15    57:6    220            (COULDBE<=2) THEN (* WHEW! *)
2782  15    57:7    228           IF KIND[EBUF^[COULDBE]]=KIND[EBUF^[COULDBE-1]] THEN
2783  15    57:8    253             FOUND:=FALSE; (* FALSE FIND... DON'T COUNT IT. *)
2784  15    57:6    257         IF (PTR+PLENGTH<=BUFCOUNT-1) AND
2785  15    57:6    268            (KIND[EBUF^[PTR+PLENGTH-1]]=KIND[EBUF^[PTR+PLENGTH]]) THEN
2786  15    57:7    302           FOUND:=FALSE; (* ANOTHER FALSE FIND *)
2787  15    57:5    306       END;
2788  15    57:1    306   UNTIL FOUND OR NOT THERE;
2789  15    57:0    316 END(* GOFORIT *);
```

```
2790  15   57:0   332
2791  15   60:0     1 PROCEDURE PUTPROMPT(LEFT,RIGHT:STRING; REPEATFACTOR:INTEGER; LORT:BOOLEAN);
2792  15   60:0     0 BEGIN
2793  15   60:1     0   PROMPTLINE:=LEFT; PROMPT;
2794  15   60:1    20   WRITE('[');
2795  15   60:1    28   IF INFINITY THEN WRITE('/') ELSE WRITE(REPEATFACTOR);
2796  15   60:1    51   WRITE(']: ');
2797  15   60:1    64   IF LORT THEN IF MODE=TOKEN THEN WRITE('L(IT') ELSE WRITE('T(OK');
2798  15   60:1   104   WRITE(RIGHT)
2799  15   60:0   113 END;
2800  15   60:0   126
2801  15   61:0     1 PROCEDURE REPLACEIT;
2802  15   61:0     1 LABEL 1;
2803  15   61:0     0 BEGIN
2804  15   61:1     0   IF VERIFY THEN
2805  15   61:2     6     BEGIN
2806  15   61:3     6       CENTERCURSOR(TRASH,MIDDLE,NOT JUSTIN);
2807  15   61:3    19       PUTPROMPT(' REPLACE','<ESC> ABORTS, ''R'' REPLACES, '' '' DOESN''T',
2808  15   61:3    72                 REPEATFACTOR-I+2,FALSE);
2809  15   61:3    82       SHOWCURSOR;
2810  15   61:3    85       CH:=GETCH;
2811  15   61:3    92       IF CH=CHR(ESC) THEN
2812  15   61:4    99         BEGIN
2813  15   61:5    99           GETLEADING; CURSOR:=MAX(CURSOR,STUFFSTART);
2814  15   61:5   111           NEXTCOMMAND; EXIT(FIND)
2815  15   61:4   117         END;
2816  15   61:3   117       IF (CH<>'R') AND (CH<>'R') THEN
2817  15   61:4   126         BEGIN
2818  15   61:5   126           REPEATFACTOR:=REPEATFACTOR+1;  (* 20-JUN-78 DON'T COUNT FALSE HITS *)
2819  15   61:5   131           GOTO 1;
2820  15   61:4   133         END;
2821  15   61:2   133     END;
2822  15   61:2   133     (* REPLACE TARGET WITH SUBSTRING *)
2823  15   61:1   133       IF SLENGTH>CURSOR-LASTPAT THEN
2824  15   61:2   142         IF SLENGTH-(CURSOR-LASTPAT)+BUFCOUNT>BUFSIZE-200 THEN
2825  15   61:3   159             BEGIN
2826  15   61:4   159               ERROR('BUFFER FULL.  ABORTING REPLACE',NONFATAL);
2827  15   61:4   196               GETLEADING; CURSOR:=MAX(CURSOR,STUFFSTART);
2828  15   61:4   208               NEXTCOMMAND; EXIT(FIND);
2829  15   61:3   214             END
2830  15   61:2   214           ELSE
```

```
2831  15   61:3   216          MOVERIGHT(EBUF^[CURSOR],EBUF^[LASTPAT+SLENGTH],BUFCOUNT-CURSOR)
2832  15   61:1   229        ELSE
2833  15   61:2   231          IF SLENGTH<CURSOR-LASTPAT THEN
2834  15   61:3   240            MOVELEFT(EBUF^[CURSOR],EBUF^[LASTPAT+SLENGTH],BUFCOUNT-CURSOR);
2835  15   61:1   253        MOVELEFT(SUBSTRING[0],EBUF^[LASTPAT],SLENGTH);
2836  15   61:1   264        IF SLENGTH<>CURSOR-LASTPAT THEN
2837  15   61:2   273          READJUST(LASTPAT,SLENGTH-(CURSOR-LASTPAT));
2838  15   61:1   284        BUFCOUNT:=BUFCOUNT+SLENGTH-(CURSOR-LASTPAT);
2839  15   61:1   295        CURSOR  :=CURSOR  +SLENGTH-(CURSOR-LASTPAT);
2840  15   61:1   306        JUSTIN:=FALSE;
2841  15   61:1   310 1:END;
2842  15   61:1   324
2843  15    8:0     0 BEGIN
2844  15    8:1     0   ALREADYSAIDGO:=FALSE; (* OK TO GO ON WITHOUT ASKING! *)
2845  15    8:1     3   JUSTIN:=TRUE;
2846  15    8:1     6   USEOLD:=FALSE;
2847  15    8:1    10   VERIFY:=FALSE;
2848  15    8:1    14   IF PAGEZERO.TOKDEF THEN MODE:=TOKEN ELSE MODE:=LITERAL;
2849  15    8:1    27   IF COMMAND=FINDC THEN
2850  15    8:2    32     PUTPROMPT(' FIND',' <TARGET>  =>',REPEATFACTOR,TRUE)
2851  15    8:1    58   ELSE
2852  15    8:2    62     PUTPROMPT(' REPLACE',' V(FY <TARG> <SUB>  =>',REPEATFACTOR,TRUE);
2853  15    8:1   102   NEEDPROMPT:=TRUE;
2854  15    8:1   106   NEXTCH; SKIP;
2855  15    8:1   110   OPTIONS;
2856  15    8:1   112   IF NOT USEOLD THEN
2857  15    8:2   118     BEGIN
2858  15    8:3   118       PARSESTRING(TARGET,TLENGTH);
2859  15    8:3   126       TDEFINED:=TRUE
2860  15    8:2   126     END;
2861  15    8:1   130   IF COMMAND=REPLACEC THEN
2862  15    8:2   135     BEGIN
2863  15    8:3   135       NEXTCH; SKIP;
2864  15    8:3   139       USEOLD:=FALSE;
2865  15    8:3   143       OPTIONS;
2866  15    8:3   145       IF NOT USEOLD THEN
2867  15    8:4   151         BEGIN
2868  15    8:5   151           PARSESTRING(SUBSTRING,SLENGTH);
2869  15    8:5   159           SDEFINED:=TRUE
2870  15    8:4   159         END
2871  15    8:2   163     END;
```

```
2872  15    8:1   163   HOME;
2873  15    8:1   166   CLEARLINE(0);
2874  15    8:1   170   IF ((COMMAND=FINDC) AND TDEFINED)
2875  15    8:1   177      OR ((COMMAND=REPLACEC) AND SDEFINED AND TDEFINED) THEN
2876  15    8:2   191     BEGIN
2877  15    8:3   191       I:=1;
2878  15    8:3   194       FOUND:=TRUE;
2879  15    8:3   197       PTR:=CURSOR;
2880  15    8:3   200       WHILE ((I<=REPEATFACTOR) OR INFINITY) AND FOUND DO
2881  15    8:4   211         BEGIN
2882  15    8:5   211           GOFORIT; (* FIND THE TARGET (HANDLES TOKEN AND LITERAL MODE) *)
2883  15    8:5   213           I:=I+1;
2884  15    8:5   218           IF FOUND THEN
2885  15    8:6   221             BEGIN
2886  15    8:7   221               CURSOR:=PTR+PLENGTH; LASTPAT:=COULDBE; (*SET UP FOR NEXT TIME*)
2887  15    8:7   229               IF COMMAND=REPLACEC THEN REPLACEIT;
2888  15    8:7   236               IF DIRECTION='<' THEN PTR:=COULDBE-1 ELSE PTR:=CURSOR;
2889  15    8:6   251             END
2890  15    8:5   251           ELSE
2891  15    8:6   253             BEGIN
2892  15    8:7   253               IF (DIRECTION='>') AND (RPAGE<FLENGTH)
2893  15    8:7   263               OR (DIRECTION='<') AND (LPAGE>0) THEN
2894  15    8:8   276                 BEGIN
2895  15    8:9   276                   IF ALREADYSAIDGO THEN CH:='Y'
2896  15    8:9   279                   ELSE
2897  15    8:0   284                     BEGIN
2898  15    8:1   284                    MSG:='END OF BUFFER ENCOUNTERED. GET MORE FROM DISK? (Y/N)';
2899  15    8:1   344                         PUTMSG;
2900  15    8:1   347                         ALREADYSAIDGO:=TRUE;
2901  15    8:1   350                         REPEAT CH:=UCLC(GETCH) UNTIL CH IN ['Y','N'];
2902  15    8:0   382                     END;
2903  15    8:9   382                   IF CH='Y' THEN
2904  15    8:0   387                     BEGIN
2905  15    8:1   387                       JUSTIN:=FALSE; (* FORCES REDISPLAY!!! *)
2906  15    8:1   390                       MSG:='FINDING'; PUTMSG;
2907  15    8:1   408                       FOUND:=TRUE;
2908  15    8:1   411                       I:=I-1; (* REALLY HAVEN'T FOUND ANYTHING *)
2909  15    8:1   416                       IF DIRECTION='>' THEN
2910  15    8:2   421                         BEGIN
2911  15    8:3   421                           CURSOR:=BUFCOUNT-1;
2912  15    8:3   426                           PUTPAGES(LEFTSTACK);
```

```
2913  15    8:3   430                          GETPAGES(RIGHTSTACK);
2914  15    8:2   434                        END
2915  15    8:1   434                      ELSE
2916  15    8:2   436                        BEGIN
2917  15    8:3   436                          CURSOR:=1;
2918  15    8:3   439                          PUTPAGES(RIGHTSTACK);
2919  15    8:3   443                          GETPAGES(LEFTSTACK)
2920  15    8:2   444                        END;
2921  15    8:1   447                      PTR:=CURSOR
2922  15    8:0   447                    END
2923  15    8:9   450                  ELSE
2924  15    8:0   452                    GOTO 1;
2925  15    8:8   454                END (* ... OR ... *)
2926  15    8:6   454              END (* IF FOUND THEN ... ELSE ... *)
2927  15    8:4   454          END; (* WHILE ... *)
2928  15    8:3   456        IF NOT FOUND THEN
2929  15    8:4   460          IF NOT( INFINITY AND (I>2) ) THEN
2930  15    8:5   470            BEGIN
2931  15    8:6   470              IF ALREADYSAIDGO THEN
2932  15    8:7   473                BEGIN (* CURSOR INVALID *)
2933  15    8:8   473                  CURSOR:=1;
2934  15    8:8   476                  JUSTIN:=FALSE;
2935  15    8:7   479                END;
2936  15    8:6   479              ERROR('PATTERN NOT IN THE FILE',NONFATAL)
2937  15    8:5   506            END;
2938  15    8:2   509      END
2939  15    8:1   509    ELSE
2940  15    8:2   511      ERROR('NO OLD PATTERN.',NONFATAL);
2941  15    8:1   533    1: GETLEADING;
2942  15    8:1   536    CURSOR:=MAX(STUFFSTART,CURSOR);
2943  15    8:1   545    CENTERCURSOR(TRASH,MIDDLE,NOT JUSTIN);
2944  15    8:1   555    SHOWCURSOR;
2945  15    8:1   558    NEXTCOMMAND
2946  15    8:0   558  END;
2947  15    8:0   584
2948  15    8:0   584  (*$TC O M M A N D   I N T E R F A C E*)
2949  15    8:0   584
2950  15    2:0     1  PROCEDURE NEXTCOMMAND;
2951  15    2:0     0  BEGIN
2952  15    2:1     0    IF NEEDPROMPT THEN
2953  15    2:2     5      BEGIN
```

```
2954  15    2:3     5       PROMPTLINE:=
2955  15    2:3     8 ' EDIT: A(DJST C(PY D(LETE F(IND I(NSRT J(MP R(PLACE Q(UIT X(CHNG Z(AP  [L.2]';
2956  15    2:3    39       PROMPT;
2957  15    2:3    92       NEEDPROMPT:=FALSE;
2958  15    2:3    96       SHOWCURSOR
2959  15    2:2    96     END;
2960  15    2:1    99   CH:=GETCH;
2961  15    2:1   106   COMMAND:=MAPTOCOMMAND(CH);
2962  15    2:0   114 END(* NEXTCOMMAND *);
2963  15    2:0   126
2964  15   62:0     1 PROCEDURE COMMANDER;
2965  15   62:0     0 BEGIN
2966  15   62:1     0   INFINITY:=FALSE;
2967  15   62:1     4   IF COMMAND=SLASHC THEN
2968  15   62:2     9     BEGIN REPEATFACTOR:=1; INFINITY:=TRUE; NEXTCOMMAND END
2969  15   62:1    18   ELSE
2970  15   62:2    20     IF COMMAND=DIGIT THEN REPEATFACTOR:=GETNUM ELSE REPEATFACTOR:=1;
2971  15   62:1    37   CASE COMMAND OF
2972  15   62:1    40     ILLEGAL: BEGIN ERRWAIT; SHOWCURSOR; NEXTCOMMAND END;
2973  15   62:1    50     REVERSEC,FORWARDC: FIXDIRECTION;
2974  15   62:1    54     BANISHC: BANISH;
2975  15   62:1    58     COPYC: COPY;
2976  15   62:1    62     DUMPC: DUMP;
2977  15   62:1    66     FINDC: FIND;
2978  15   62:1    70     INSERTC: INSERTIT;
2979  15   62:1    74     JUMPC: JUMP;
2980  15   62:1    78     LISTC: NEXTCOMMAND; (* NOT YET, DEPENDS ON TERAK PAN *)
2981  15   62:1    82     MACRODEFC: DEFMACRO;
2982  15   62:1    86     NEXTC: NEXT;
2983  15   62:1    90     QUITC: ; (* EXIT HANDLED IN OUTER BLOCK *)
2984  15   62:1    92     REPLACEC: FIND;
2985  15   62:1    96     SETC: SETSTUFF;
2986  15   62:1   100     VERIFYC: VERIFY;
2987  15   62:1   104     XECUTEC: XMACRO;
2988  15   62:1   108     ZAPC: ZAPIT;
2989  15   62:1   112     EQUALC: BEGIN
2990  15   62:3   112       CURSOR:=LASTPAT;
2991  15   62:3   115       GETLEADING;
2992  15   62:3   118       CURSOR:=MAX(CURSOR,STUFFSTART);
2993  15   62:3   127       CENTERCURSOR(TRASH,MIDDLE,FALSE);
2994  15   62:3   137       SHOWCURSOR; NEXTCOMMAND
```

```
2995  15   62:2   140     END;
2996  15   62:1   144     ADJUSTC,DELETEC,PARAC,UP,DOWN,LEFT,RIGHT,ADVANCE,TAB,SPACE: MOVEIT
2997  15   62:1   144   END (* BIG LONG CASE STATEMENT *);
2998  15   62:0   216 END (* COMMANDER *);
2999  15   62:0   230
3000  15    1:0     0 BEGIN (* EDITCORE *)
3001  15    1:1     0   NEXTCOMMAND;
3002  15    1:1     2   WHILE COMMAND<>QUITC DO COMMANDER
3003  15    1:0     7 END;
3004  15    1:0    26
3005  15    1:0    26
3006  15    1:0    26 (*$TM I S C.  P R O C E D U R E S  (INCL. SCREEN CONTROL) *)
3007  15    1:0    26
3008   1   12:D     3 FUNCTION MIN(* (A,B:INTEGER):INTEGER *);
3009   1   12:0     0 BEGIN
3010   1   12:1     0   IF A<B THEN MIN:=A ELSE MIN:=B
3011   1   12:0    10 END;
3012   1   12:0    26
3013   1   13:D     3 FUNCTION MAX (*(A,B:INTEGER):INTEGER*);
3014   1   13:0     0 BEGIN
3015   1   13:1     0   IF A>B THEN MAX:=A ELSE MAX:=B
3016   1   13:0    10 END;
3017   1   13:0    26
3018   1    4:D     3 FUNCTION GETCH(*:CHAR*);
3019   1    4:D     3 VAR GCH: CHAR;
3020   1    4:0     0 BEGIN
3021   1    4:1     0   READ(KEYBOARD,GCH);
3022   1    4:1     8   IF EOLN(KEYBOARD) THEN GCH:=CHR(EOL);
3023   1    4:1    21   GETCH:=GCH;
3024   1    4:0    24 END;
3025   1    4:0    36
3026   1    8:D     3 FUNCTION MAPTOCOMMAND(* (CH:CHAR): COMMANDS *);
3027   1    8:0     0 BEGIN
3028   1    8:1     0   IF (CH=SYSCOM^.CRTCTRL.ESCAPE) AND (CH<>CHR(0)) THEN
3029   1    8:2    16     BEGIN
3030   1    8:3    16       CH:=GETCH;
3031   1    8:3    22       IF CH=SYSCOM^.CRTINFO.LEFT THEN MAPTOCOMMAND:=LEFT
3032   1    8:3    34       ELSE
3033   1    8:4    39         IF CH=SYSCOM^.CRTINFO.RIGHT THEN MAPTOCOMMAND:=RIGHT
3034   1    8:4    51         ELSE
3035   1    8:5    56           IF CH=SYSCOM^.CRTINF  UP THEN MAPTOCOMMAND:=UP
```

```
3036  1   8:5   68            ELSE
3037  1   8:6   73              IF CH=SYSCOM^.CRTINFO.DOWN THEN MAPTOCOMMAND:=DOWN
3038  1   8:6   85              ELSE
3039  1   8:7   90                MAPTOCOMMAND:=ILLEGAL
3040  1   8:2   90      END
3041  1   8:1   93    ELSE
3042  1   8:2   95      MAPTOCOMMAND:=TRANSLATE[CH];
3043  1   8:0  104  END;
3044  1   8:0  116
3045  1   9:0    3  FUNCTION UCLC(*(CH:CHAR):CHAR*); (* MAP LOWER CASE TO UPPER CASE *)
3046  1   9:0    0  BEGIN
3047  1   9:1    0    IF CH IN ['A'..'Z'] THEN UCLC:=CHR(ORD(CH)-32) ELSE UCLC:=CH
3048  1   9:0   31  END;
3049  1   9:0   46
3050  1  14:0    1  PROCEDURE CONTROL(*CH:CTYPE*);
3051  1  14:0    2  (* BASED ON THE PARAMETER PASSED, USE CRTCTRL TO PUT OUT THE
3052  1  14:0    2     APPROPRIATE CONTROL CODE FOR THE HOST TERMINAL *)
3053  1  14:0    0  BEGIN
3054  1  14:1    0    WITH SYSCOM^.CRTCTRL DO
3055  1  14:2    7      BEGIN
3056  1  14:3    7        IF ESCAPE<>CHR(0) THEN WRITE(ESCAPE);
3057  1  14:3   26        CASE CH OF
3058  1  14:3   29          FS:     WRITE(NDFS);
3059  1  14:3   44          GOHOME: WRITE(HOME);
3060  1  14:3   57          ETOEOL: WRITE(ERASEEOL);
3061  1  14:3   72          ETOEOS: WRITE(ERASEEOS);
3062  1  14:3   87          US:     WRITE(RLF)
3063  1  14:3  100        END
3064  1  14:2  120      END
3065  1  14:0  120  END;
3066  1  14:0  132
3067  1  14:0  132  (* LOOK AT ME! LOOK AT ME! LOOK AT ME! LOOK AT ME! LOOK AT ME! LOOK AT ME!  *)
3068  1   5:0    1  PROCEDURE CLEARSCREEN;
3069  1   5:0    1  (* SET THE SCREEN TO ALL BLANKS AND LEAVE THE CURSOR IN THE UPPER LEFT-HAND
3070  1   5:0    1     CORNER (0,0).  NOTE THAT THE CONTROL CODE FOR THIS OPERATION IS HARD-
3071  1   5:0    1     WIRED (I.E. IT DOESN'T GO THROUGH SYSCOM), AND THUS ENTAILS A RECOMP-
3072  1   5:0    1     ILATION TO CHANGE TERMINALS.  P.S. 12 IS A FF.  *)
3073  1   5:0    0  BEGIN
3074  1   5:1    0    WRITE(CHR(12))
3075  1   5:0    8  END;
3076  1   5:0   20
```

```
3077   1     7:D     1 PROCEDURE CLEARLINE(*Y:INTEGER*);
3078   1     7:0     2 (* IF YOUR TERMINAL HAS AN ERASELINE CAPABILITY; THAT IS A CONTROL CODE
3079   1     7:D     2    THAT WILL CLEAR THE LINE THE CURSOR IS ON, AND LEAVE THE CURSOR AT
3080   1     7:0     2    THE FIRST COLUMN (0,Y) THEN SUBSTITUTE THIS CODE WITH A SINGLE CHARACTER
3081   1     7:D     2    WRITE *)
3082   1     7:0     0 BEGIN
3083   1     7:0     0   (*
3084   1     7:0     0   IF Y<>SCREENHEIGHT THEN UNITWRITE(2,BLANKAREA,SCREENWIDTH+1)
3085   1     7:0     0   ELSE UNITWRITE(2,BLANKAREA,SCREENWIDTH);
3086   1     7:0     0   GOTOXY(0,Y);
3087   1     7:0     0   *)
3088   1     7:1     0   GOTOXY(0,Y); CONTROL(ETOEOL);
3089   1     7:0     8 END;
3090   1     7:0    20
3091   1    15:D     1 PROCEDURE PUTMSG;
3092   1    15:0     0 BEGIN
3093   1    15:1     0   CONTROL(GOHOME);
3094   1    15:1     3   CLEARLINE(0);
3095   1    15:1     6   SAVETOP:=MSG;
3096   1    15:1    14   WRITE(MSG);
3097   1    15:0    24 END;
3098   1    15:0    36
3099   1    16:0     0 PROCEDURE HOME; BEGIN CONTROL(GOHOME) END;
3100   1    16:0    16
3101   1     3:D     1 PROCEDURE ERASETOEOL(*X,LINE:INTEGER*);
3102   1     3:0     0 BEGIN
3103   1     3:0     0   (*
3104   1     3:0     0   IF X=0 THEN CLEARLINE(LINE)
3105   1     3:0     0   ELSE
3106   1     3:0     0     BEGIN
3107   1     3:0     0       IF LINE=SCREENHEIGHT THEN
3108   1     3:0     0         UNITWRITE(2,BLANKAREA,SCREENWIDTH-X)
3109   1     3:0     0       ELSE
3110   1     3:0     0         UNITWRITE(2,BLANKAREA,SCREENWIDTH-X+1)
3111   1     3:0     0     END;
3112   1     3:0     0   GOTOXY(X,LINE);
3113   1     3:0     0   *)
3114   1     3:1     0   CONTROL(ETOEOL);
3115   1     3:0     3 END;
3116   1     3:0    16
3117   1     6:D     1 PROCEDURE ERASEOS(*X,LINE*);
```

```
3118   1    6:0     3 VAR I: INTEGER;
3119   1    6:0     0 BEGIN
3120   1    6:0     0   (*
3121   1    6:0     0   ERASETOEOL(X,LINE);
3122   1    6:0     0   FOR I:=LINE+1 TO SCREENHEIGHT DO BEGIN WRITELN; CLEARLINE(I) END;
3123   1    6:0     0   *)
3124   1    6:1     0   CONTROL(ETOEOS);
3125   1    6:0     3 END;
3126   1    6:0    16
3127   1   10:0     1 PROCEDURE PROMPT;
3128   1   10:0     0 BEGIN
3129   1   10:1     0   PROMPTLINE[1]:=DIRECTION;
3130   1   10:1     6   SAVETOP:=PROMPTLINE;
3131   1   10:1    14   CONTROL(GOHOME);
3132   1   10:1    17   CLEARLINE(0);
3133   1   10:1    20   WRITE(PROMPTLINE)
3134   1   10:0    30 END;
3135   1   10:0    42
3136   1   17:0     1 PROCEDURE ERRWAIT;
3137   1   17:0     0 BEGIN
3138   1   17:1     0   WRITE(CHR(BELL));
3139   1   17:1     8   PROMPT;
3140   1   17:0    10 END;
3141   1   17:0    22
3142   1   18:0     1 PROCEDURE BLANKCRT(*Y: INTEGER*);
3143   1   18:0     0 BEGIN
3144   1   18:0     0   (*
3145   1   18:0     0   IF Y=1 THEN
3146   1   18:0     0     BEGIN
3147   1   18:0     0       CLEARSCREEN;
3148   1   18:0     0       WRITELN(SAVETOP)
3149   1   18:0     0     END
3150   1   18:0     0   ELSE
3151   1   18:0     0     BEGIN
3152   1   18:0     0       GOTOXY(0,Y);
3153   1   18:0     0       ERASEOS(0,Y);
3154   1   18:0     0     END;
3155   1   18:0     0   *)
3156   1   18:1     0   GOTOXY(0,Y);
3157   1   18:1     5   CONTROL(ETOEOS)
3158   1   18:0     6 END;
```

```
3159  1   18:0    20
3160  1    2:0     1 PROCEDURE ERROR(*S: STRING;HOWBAD: ERRORTYPE*);
3161  1    2:0     0 BEGIN
3162  1    2:1     0   UNITCLEAR(1); (* THROW AWAY ALL CHARACTERS QUEUED UP *)
3163  1    2:1     8   IF HOWBAD=FATAL THEN
3164  1    2:2    13     BLANKCRT(1)
3165  1    2:1    14   ELSE
3166  1    2:2    18     BEGIN HOME; CLEARLINE(0) END;
3167  1    2:1    23   WRITE('ERROR: ',S);
3168  1    2:1    49   IF HOWBAD=FATAL THEN
3169  1    2:2    54     EXIT(EDITOR)
3170  1    2:1    58   ELSE
3171  1    2:2    60     BEGIN
3172  1    2:3    60       WRITE('  PLEASE PRESS <SPACEBAR> TO CONTINUE.');
3173  1    2:3   108       REPEAT UNTIL GETCH=' '; NEEDPROMPT:=TRUE
3174  1    2:2   116     END;
3175  1    2:0   120 END;
3176  1    2:0   134
3177  1    2:0   134 (*$TU T I L I T Y   P R O C E D U R E S*)
3178  1    2:0   134
3179  1   19:D     3 FUNCTION LEADBLANKS(* (PTR: PTRTYPE; VAR BYTES: INTEGER): INTEGER *);
3180  1   19:D     5 (* ON ENTRY-
3181  1   19:D     5       PTR POINTS TO THE BEGINNING OF A LINE
3182  1   19:D     5    ON EXIT-
3183  1   19:D     5       FUNCTION RETURNS THE NUMBER OF LEADING BLANKS ON THAT LINE.
3184  1   19:D     5       BYTES HAS THE OFFSET INTO THE LINE OF THE FIRST NON-BLANK CHARACTER *)
3185  1   19:D     5 VAR
3186  1   19:D     5   OLDPTR: PTRTYPE;
3187  1   19:D     6   INDENT: INTEGER;
3188  1   19:0     0 BEGIN
3189  1   19:1     0   OLDPTR:=PTR; INDENT:=0;
3190  1   19:1     6   WHILE ORD(EBUF^[PTR]) IN [HT,SP,DLE] DO
3191  1   19:2    22     BEGIN
3192  1   19:3    22       IF EBUF^[PTR]=CHR(DLE) THEN
3193  1   19:4    29         BEGIN PTR:=PTR+1; INDENT:=INDENT+ORD(EBUF^[PTR])-32 END
3194  1   19:3    43       ELSE
3195  1   19:4    45         IF ORD(EBUF^[PTR])=SP THEN INDENT:=INDENT+1
3196  1   19:4    53         ELSE
3197  1   19:5    59           (*HT*) INDENT:=((INDENT DIV 8)+1)*8;  (* KLUDGE FOR COLUMNAR TAB! *)
3198  1   19:3    68       PTR:=PTR+1
3199  1   19:2    69     END;
```

```
3200   1   19:1    75    BYTES:=PTR-OLDPTR;
3201   1   19:1    80    LEADBLANKS:=INDENT;
3202   1   19:0    83  END(*LEADBLANKS*);
3203   1   19:0    98
3204   1   11:D     1  PROCEDURE REDISPLAY;
3205   1   11:D     1  (* DO A TOTAL UPDATE OF THE SCREEN.  NOTE THAT THIS CODE IS PARTIALLY A
3206   1   11:D     1     DUPLICATE OF LINEOUT/UPSCREEN FOR REASONS OF SPEED.  THIS PROCEDURE IS
3207   1   11:D     1     CALLED ONLY FROM CENTERCURSOR *)
3208   1   11:D     1  VAR
3209   1   11:D     1    LINEDIST,EOLDIST,LINE: INTEGER;
3210   1   11:D     4    PTR: PTRTYPE;
3211   1   11:D     5    T: PACKED ARRAY [0..MAXSW] OF CHAR;
3212   1   11:0     0  BEGIN
3213   1   11:1     0    BLANKCRT(1);
3214   1   11:1     3    LINE:=1;
3215   1   11:1     6    PTR:=LINE1PTR;
3216   1   11:1    11    REPEAT
3217   1   11:2    11      BLANKS:=MIN(LEADBLANKS(PTR,BYTES),SCREENWIDTH);
3218   1   11:2    25      GOTOXY(BLANKS,LINE);
3219   1   11:2    30      PTR:=PTR+BYTES;
3220   1   11:2    35      EOLDIST:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR]);
3221   1   11:2    47      LINEDIST:=MAX(0,MIN(EOLDIST,SCREENWIDTH-BLANKS+1));
3222   1   11:2    64      MOVELEFT(EBUF^[PTR],T[0],LINEDIST);
3223   1   11:2    72      IF EBUF^[PTR+LINEDIST]<>CHR(EOL) THEN (* LINE TRUNCATION *)
3224   1   11:3    81        T[MAX(0,LINEDIST-1)]:='!';
3225   1   11:2    93      WRITE(T:LINEDIST);
3226   1   11:2   103      PTR:=PTR+EOLDIST+1; LINE:=LINE+1
3227   1   11:1   111    UNTIL (LINE>SCREENHEIGHT) OR (PTR>=BUFCOUNT)
3228   1   11:0   121  END;
3229   1   11:0   138
3230   1    1:D     1  PROCEDURE CENTERCURSOR
3231   1   20:D     4  (*VAR LINE: INTEGER; LINESUP: INTEGER; NEWSCREEN: BOOLEAN*);
3232   1   20:D     4  (* FIGURE OUT IF THE CURSOR IS STILL ON THE SCREEN.  IF IT IS, AND
3233   1   20:D     4     NEWSCREEN IS FALSE, THEN NO REDISPLAY IS DONE.  OTHERWISE AN ATTEMPT
3234   1   20:D     4     IS MADE TO POSITION THE CURSOR AT LINE "LINESUP".  LINE IS THEN UPDATED
3235   1   20:D     4     TO THE ACTUAL LINE THE CURSOR WAS FORCED TO. *)
3236   1   20:D     4  VAR
3237   1   20:D     4    MARK: INTEGER;
3238   1   20:D     5    PTR: PTRTYPE;
3239   1   20:0     0  BEGIN
3240   1   20:1     0    IF EBUF^[CURSOR]=CHR(EOL) THEN PTR:=CURSOR ELSE PTR:=CURSOR+1;
```

```
3241   1   20:1    17   LINE:=0;
3242   1   20:1    20   REPEAT
3243   1   20:2    20     PTR:=PTR-1;
3244   1   20:2    25     PTR:=SCAN(-MAXCHAR,=CHR(EOL),EBUF^[PTR])+PTR;
3245   1   20:2    40     LINE:=LINE+1;
3246   1   20:2    46     IF LINE=LINESUP THEN MARK:=PTR;
3247   1   20:1    55   UNTIL (LINE>SCREENHEIGHT) OR ((LINE1PTR=PTR+1) AND NOT NEWSCREEN) OR (PTR<1);
3248   1   20:1    76   IF LINE>SCREENHEIGHT THEN (* OFF THE SCREEN *)
3249   1   20:2    82     BEGIN LINE1PTR:=MARK+1; REDISPLAY; LINE:=LINESUP END
3250   1   20:1    93   ELSE
3251   1   20:2    95     IF LINE1PTR=PTR+1 THEN
3252   1   20:3   104       BEGIN
3253   1   20:4   104         IF NEWSCREEN THEN REDISPLAY
3254   1   20:3   107       END
3255   1   20:2   109     ELSE
3256   1   20:3   111       BEGIN
3257   1   20:4   111         LINE1PTR:=1; REDISPLAY
3258   1   20:3   115       END;
3259   1   20:0   117 END;
3260   1   20:0   132
3261   1   21:D     1 PROCEDURE FINDXY(*VAR INDENT,LINE: INTEGER*);
3262   1   21:D     3 VAR
3263   1   21:D     3   I,LEAD: INTEGER;
3264   1   21:D     5   PTR,EOLPTR: PTRTYPE;
3265   1   21:0     0 BEGIN
3266   1   21:0     0   (* PLACE CRT CURSOR ON THE SCREEN AT THE POSITION CORRESPONDING
3267   1   21:0     0      TO THE LOGICAL CURSOR. *)
3268   1   21:1     0   LINE:=1;
3269   1   21:1     3   PTR:=LINE1PTR;
3270   1   21:1     8   EOLPTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR])+PTR;
3271   1   21:1    22   WHILE EOLPTR<CURSOR DO
3272   1   21:2    27     BEGIN
3273   1   21:3    27       LINE:=LINE+1; PTR:=EOLPTR+1; (* SET UP FOR THE NEXT LINE *)
3274   1   21:3    38       EOLPTR:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR])+PTR
3275   1   21:2    48     END;
3276   1   21:2    54   (* NOW FIND THE INDENTATION ON THAT LINE OF THE CURSOR *)
3277   1   21:1    54   LEAD:=LEADBLANKS(PTR,I);
3278   1   21:1    63   INDENT:=MIN(SCREENWIDTH,(LEAD-I)+(CURSOR-PTR));
3279   1   21:1    77                              (* (EXTRA SPACES) + (OFFSET INTO LINE) *)
3280   1   21:0    77 END;(* FINDXY *)
3281   1   21:0    92
```

```
3282   1   22:0     1 PROCEDURE SHOWCURSOR;
3283   1   22:0     1 VAR
3284   1   22:0     1   X,Y: INTEGER;
3285   1   22:0     0 BEGIN
3286   1   22:1     0   FINDXY(X,Y);
3287   1   22:1     6   GOTOXY(X,Y)
3288   1   22:0    11 END(* SHOWCURSOR *);
3289   1   22:0    24
3290   1   23:0     3 FUNCTION GETNUM(*:INTEGER*);
3291   1   23:0     3 VAR
3292   1   23:0     3   N: INTEGER;
3293   1   23:0     4   OVERFLOW: BOOLEAN;
3294   1   23:0     0 BEGIN
3295   1   23:1     0   N:=0;
3296   1   23:1     3   OVERFLOW:=FALSE;
3297   1   23:1     6   IF NOT (CH IN ['0'..'9']) THEN N:=1
3298   1   23:1    23   ELSE
3299   1   23:2    28     REPEAT
3300   1   23:3    28       IF N > 1000 THEN OVERFLOW:=TRUE
3301   1   23:3    35       ELSE
3302   1   23:4    40         BEGIN
3303   1   23:5    40           N:=N*10+ORD(CH)-ORD('0');
3304   1   23:5    49           CH:=GETCH
3305   1   23:4    49         END
3306   1   23:2    55     UNTIL (NOT (CH IN ['0'..'9'])) OR OVERFLOW;
3307   1   23:1    73   IF OVERFLOW THEN
3308   1   23:2    76     BEGIN
3309   1   23:3    76       ERROR('REPEATFACTOR > 10,000',NONFATAL);
3310   1   23:3   103       GETNUM:=0;
3311   1   23:2   106     END
3312   1   23:1   106   ELSE
3313   1   23:2   108     GETNUM:=N;
3314   1   23:1   111   COMMAND:=MAPTOCOMMAND(CH); (* TAKES CH AND MAPS IT TO A COMMAND *)
3315   1   23:0   118 END;
3316   1   23:0   132
3317   1   24:0     1 PROCEDURE GETLEADING;
3318   1   24:0     0 BEGIN
3319   1   24:0     0   (* SETS:
3320   1   24:0     0         LINESTART ......... A POINTER TO THE BEGINNING OF THE LINE
3321   1   24:0     0         STUFFSTART ........ A POINTER TO THE BEGINNING OF THE TEXT ON THE LINE
3322   1   24:0     0         BYTES ............. THE NUMBER OF BYTES BETWEEN LINESTART AND
```

```
3323   1   24:0     0                                STUFFSTART
3324   1   24:0     0          BLANKS ..... THE INDENTATION OF THE LINE    *)
3325   1   24:1     0   LINESTART:=CURSOR;
3326   1   24:1     3   IF EBUF^[LINESTART]=CHR(EOL) THEN LINESTART:=LINESTART-1; (* FOR SCAN! *)
3327   1   24:1    15   LINESTART:=SCAN(-MAXCHAR,=CHR(EOL),EBUF^[LINESTART])+LINESTART+1;
3328   1   24:1    32   BLANKS:=LEADBLANKS(LINESTART,BYTES);
3329   1   24:1    41   STUFFSTART:=LINESTART+BYTES
3330   1   24:0    42 END (* GETLEADING *);
3331   1   24:0    58
3332   1   25:0     3 FUNCTION OKTODEL (* (CURSOR,ANCHOR: PTRTYPE):BOOLEAN *) ;
3333   1   25:0     0 BEGIN
3334   1   25:1     0   IF ABS(CURSOR-ANCHOR)>(BUFSIZE-BUFCOUNT)+10 THEN
3335   1   25:2    12     BEGIN
3336   1   25:3    12       MSG:=
3337   1   25:3    15 'THERE IS NO ROOM TO COPY THE DELETION.  DO YOU WISH TO DELETE ANYWAY? (Y/N)';
3338   1   25:3    95       PUTMSG;
3339   1   25:3    97       IF UCLC(GETCH)='Y' THEN OKTODEL:=TRUE ELSE OKTODEL:=FALSE;
3340   1   25:2   117     END
3341   1   25:1   117   ELSE
3342   1   25:2   119     BEGIN
3343   1   25:2   119       (* COPYLINE IS SET BY THE CALLER *)
3344   1   25:3   119       COPYOK:=TRUE; COPYLENGTH:=ABS(CURSOR-ANCHOR);
3345   1   25:3   130       COPYSTART:=BUFSIZE-COPYLENGTH+1;
3346   1   25:3   140       MOVELEFT(EBUF^[MIN(CURSOR,ANCHOR)],EBUF^[COPYSTART],COPYLENGTH);
3347   1   25:3   156       OKTODEL:=TRUE
3348   1   25:2   156     END;
3349   1   25:0   159 END;
3350   1   25:0   172
3351   1   25:0   172
3352   1   26:0     1 PROCEDURE LINEOUT(*VAR PTR:PTRTYPE; BYTES,BLANKS,LINE:INTEGER*);
3353   1   26:0     5 (* WRITE A LINE OUT *)
3354   1   26:0     5 VAR
3355   1   26:0     5   LINEDIST,EOLDIST: INTEGER;
3356   1   26:0     7   T: PACKED ARRAY [0..MAXSW] OF CHAR;
3357   1   26:0     0 BEGIN
3358   1   26:1     0   GOTOXY(BLANKS,LINE);
3359   1   26:1     5   PTR:=PTR+BYTES;
3360   1   26:1    11   EOLDIST:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[PTR]);
3361   1   26:1    24   LINEDIST:=MAX(0,MIN(EOLDIST,SCREENWIDTH-BLANKS+1));
3362   1   26:1    41   MOVELEFT(EBUF^[PTR],T[0],LINEDIST);
3363   1   26:1    50   IF EBUF^[PTR+LINEDIST]<>CHR(EOL) THEN (* LINE TRUNCATION *)
```

```
3364   1   26:2    60      BEGIN
3365   1   26:3    60        LINEDIST:=MAX(LINEDIST,1);
3366   1   26:3    68        T[LINEDIST-1]:='!';
3367   1   26:2    75      END;
3368   1   26:1    75    WRITE(T:LINEDIST);
3369   1   26:1    85    PTR:=PTR+EOLDIST+1
3370   1   26:0    90  END;
3371   1   26:0   106
3372   1   27:D     1  PROCEDURE UPSCREEN(*FIRSTLINE,WHOLESCREEN: BOOLEAN; LINE: INTEGER*);
3373   1   27:D     4  (* ZAP, INSERT AND DELETE CALL THIS PROCEDURE TO UPDATE (POSSIBLY PARTIALLY)
3374   1   27:D     4     THE SCREEN.  FIRSTLINE MEANS ONLY THE LINE THAT THE CURSOR IS ON NEED
3375   1   27:D     4     BE UPDATED.  WHOLESCREEN MEANS THAT EVERYTHING MUST BE UPDATED.  IF
3376   1   27:D     4     NEITHER OF THESE IS TRUE THEN ONLY THE PART OF THE SCREEN THAT'S AFTER
3377   1   27:D     4     THE CURSOR IS UPDATED *)
3378   1   27:D     4  VAR
3379   1   27:D     4    PTR: PTRTYPE;
3380   1   27:D     5
3381   1   27:0     0  BEGIN (* UPSCREEN *)
3382   1   27:1     0    IF FIRSTLINE THEN
3383   1   27:2     3      BEGIN
3384   1   27:3     3        GETLEADING;
3385   1   27:3     5        GOTOXY(0,LINE); ERASETOEOL(0,LINE); (* CLEAN THE LINE *)
3386   1   27:3    14        LINEOUT(LINESTART,BYTES,BLANKS,LINE) (* JUST THIS LINE *)
3387   1   27:2    19      END
3388   1   27:1    21    ELSE
3389   1   27:2    23      IF WHOLESCREEN THEN
3390   1   27:3    26        CENTERCURSOR(TRASH,MIDDLE,TRUE)
3391   1   27:2    33      ELSE (* ONLY UPDATE THE PART OF THE SCREEN AFTER THE CURSOR *)
3392   1   27:3    37        BEGIN
3393   1   27:4    37          GOTOXY(0,LINE); ERASEOS(0,LINE);
3394   1   27:4    46          GETLEADING;
3395   1   27:4    48          PTR:=LINESTART;
3396   1   27:4    51          REPEAT
3397   1   27:5    51            BLANKS:=MIN(LEADBLANKS(PTR,BYTES),SCREENWIDTH);
3398   1   27:5    65            LINEOUT(PTR,BYTES,BLANKS,LINE); (* WRITES OUT THE LINE AT PTR *)
3399   1   27:5    72            LINE:=LINE+1
3400   1   27:4    73          UNTIL (LINE>SCREENHEIGHT) OR (PTR>=BUFCOUNT)
3401   1   27:3    83        END;
3402   1   27:0    86  END;
3403   1   27:0   100
3404   1   28:D     1  PROCEDURE READJUST(*CURSOR:PTRTYPE; DELTA: INTEGER*);
```

```
3405   1   28:0     3 (* IF DELTA<0 THEN MOVE ALL AFFECTED MARKERS TO CURSOR.  ALSO ADJUST ALL
3406   1   28:0     3    MARKERS >= CURSOR BY DELTA *)
3407   1   28:0     3 VAR
3408   1   28:0     3   I: INTEGER;
3409   1   28:0     0 BEGIN
3410   1   28:1     0   WITH PAGEZERO DO
3411   1   28:2     0     FOR I:=0 TO COUNT-1 DO
3412   1   28:3    15       IF PAGEN[I]=-1 THEN
3413   1   28:4    27         IF POFFSET[I]>=CURSOR THEN
3414   1   28:5    38           BEGIN
3415   1   28:6    38             POFFSET[I]:=MAX(POFFSET[I]+DELTA,CURSOR);
3416   1   28:5    59           END;
3417   1   28:1    66   IF (COPYSTART>=CURSOR) AND (COPYSTART<BUFCOUNT) THEN
3418   1   28:2    79     COPYSTART:=MAX(COPYSTART+DELTA,CURSOR);
3419   1   28:0    92 END;
3420   1   28:0   106
3421   1   29:0     1 PROCEDURE THEFIXER(*PARAPTR:PTRTYPE;RFAC:INTEGER;WHOLE:BOOLEAN*);
3422   1   29:0     4 (* PARAPTR POINTS SOMEWHERE IN A PARAGRAPH.  IF WHOLE IS TRUE THEN THE
3423   1   29:0     4    ENTIRE PARAGRAPH IS FILLED, OTHERWISE ONLY THAT DIRECTLY AFTER THE CURSOR
3424   1   29:0     4    IS FILLED.  RFAC, WHEN IMPLEMENTED WILL TELL HOW MANY PARAGRAPHS TO BE
3425   1   29:0     4    FILLED.  NOTE: A PARAGRAPH IS DEFINED AS LINES OF TEXT DELIMITED BY A LINE
3426   1   29:0     4    WITH NO TEXT ON IT WHATSOEVER, OR A LINE OF A TEXT WHOSE FIRST CHARACTER IS
3427   1   29:0     4    RUNOFFCH *)
3428   1   29:0     4
3429   1   29:0     4 VAR
3430   1   29:0     4   SAVE,PTR,WPTR: INTEGER;
3431   1   29:0     7   WLENGTH,X: INTEGER;
3432   1   29:0     9   DONE: BOOLEAN;
3433   1   29:0     0 BEGIN
3434   1   29:1     0   WITH PAGEZERO DO
3435   1   29:2     0     BEGIN
3436   1   29:3     0       SAVE:=CURSOR;
3437   1   29:3     3       CURSOR:=PARAPTR;
3438   1   29:3     6       GETLEADING;
3439   1   29:3     8       IF EBUF^[STUFFSTART] IN [CHR(EOL),RUNOFFCH] THEN EXIT(THEFIXER);
3440   1   29:3    25       IF WHOLE THEN (* SCAN BACKWARDS FOR THE BEGINNING OF THE PARAGRAPH *)
3441   1   29:4    28         BEGIN
3442   1   29:5    28           REPEAT
3443   1   29:6    28             CURSOR:=LINESTART-1;
3444   1   29:6    33             GETLEADING
3445   1   29:5    33           UNTIL (LINESTART<=1) OR (EBUF^[STUFFSTART] IN [RUNOFFCH,CHR(EOL)]);
```

```
3446  1  29:5   52          IF EBUF^[STUFFSTART] IN [RUNOFFCH,CHR(EOL)] THEN
3447  1  29:6   65            PTR:=CURSOR+1
3448  1  29:5   66          ELSE
3449  1  29:6   72            PTR:=1;
3450  1  29:5   75          X:=PARAMARGIN;
3451  1  29:4   80        END
3452  1  29:3   80      ELSE
3453  1  29:4   82        BEGIN
3454  1  29:5   82          PTR:=LINESTART;
3455  1  29:5   85          IF BLANKS=PARAMARGIN THEN X:=PARAMARGIN ELSE X:=LMARGIN
3456  1  29:4   99        END;
3457  1  29:3  104      CURSOR:=BUFSIZE-(BUFCOUNT-PTR)+1; (* SPLIT THE BUFFER *)
3458  1  29:3  113      MOVERIGHT(EBUF^[PTR],EBUF^[CURSOR],BUFCOUNT-PTR);
3459  1  29:3  122      (* NOW DRIBBLE BACK THE (REST OF THE) PARAGRAPH *)
3460  1  29:3  122      EBUF^[PTR]:=CHR(DLE);
3461  1  29:3  126      EBUF^[PTR+1]:=CHR(X+32);
3462  1  29:3  134      PTR:=PTR+2;
3463  1  29:3  139      EBUF^[CURSOR-1]:=CHR(EOL); (* SENTINEL FOR GETLEADING *)
3464  1  29:3  145      DONE:=FALSE;
3465  1  29:3  148      REPEAT
3466  1  29:4  148        WHILE EBUF^[CURSOR] IN [CHR(HT),CHR(SP),CHR(DLE)] DO
3467  1  29:5  162          IF EBUF^[CURSOR]=CHR(DLE) THEN CURSOR:=CURSOR+2 ELSE CURSOR:=CURSOR+1;
3468  1  29:4  183        WPTR:=CURSOR;
3469  1  29:4  186        (* SKIP OVER A TOKEN *)
3470  1  29:4  186        WHILE NOT (EBUF^[CURSOR] IN [CHR(EOL),' ','-']) DO CURSOR:=CURSOR+1;
3471  1  29:4  213        (* SPECIAL CASES FOR ".<SP><SP>" AND "-<SP>" *)
3472  1  29:4  213        IF EBUF^[CURSOR]='-' THEN IF EBUF^[CURSOR+1]=' ' THEN CURSOR:=CURSOR+1;
3473  1  29:4  234        IF (EBUF^[CURSOR-1] IN ['.','?','!','"']) THEN IF
3474  1  29:5  254           (EBUF^[CURSOR]=' ') AND (EBUF^[CURSOR+1]=' ') THEN CURSOR:=CURSOR+1;
3475  1  29:4  274        WLENGTH:=CURSOR-WPTR+1; (* INCLUDING THE DELIMITER *)
3476  1  29:4  281        IF (X+WLENGTH>RMARGIN) OR (RMARGIN-LMARGIN+1<=WLENGTH) THEN
3477  1  29:5  302          BEGIN
3478  1  29:6  302            IF EBUF^[PTR-1]=' ' THEN PTR:=PTR-1;
3479  1  29:6  316            EBUF^[PTR]:=CHR(EOL); EBUF^[PTR+1]:=CHR(DLE);
3480  1  29:6  326            EBUF^[PTR+2]:=CHR(LMARGIN+32);
3481  1  29:6  336            PTR:=PTR+3;
3482  1  29:6  341            X:=LMARGIN
3483  1  29:5  341          END;
3484  1  29:4  346        CURSOR:=CURSOR+1;
3485  1  29:4  351        MOVELEFT(EBUF^[WPTR],EBUF^[PTR],WLENGTH);
3486  1  29:4  358        IF EBUF^[CURSOR-1]=CHR(EOL) THEN
```

```
3487   1   29:5   367            BEGIN
3488   1   29:6   367              IF EBUF^[CURSOR]=CHR(0) THEN DONE:=TRUE
3489   1   29:6   374              ELSE
3490   1   29:7   379                BEGIN
3491   1   29:8   379                  GETLEADING;
3492   1   29:8   381                  DONE:=(EBUF^[STUFFSTART]=CHR(EOL))
3493   1   29:8   386                        OR (EBUF^[STUFFSTART]=RUNOFFCH);
3494   1   29:8   396                  (* THE LAST TRANSFER WILL MOVE
3495   1   29:8   396                     OVER THE <EOL> FOR THE PARAGRAPH *)
3496   1   29:8   396                  IF NOT DONE THEN
3497   1   29:9   400                    BEGIN
3498   1   29:0   400                      EBUF^[PTR+WLENGTH-1]:=' ';
3499   1   29:0   408                      (* IF <EOL> <SP>, MAP TO ONE SPACE ONLY *)
3500   1   29:0   408                      IF EBUF^[CURSOR-2]=' ' THEN PTR:=PTR-1;
3501   1   29:9   422                    END
3502   1   29:7   422                END
3503   1   29:5   422              END;
3504   1   29:4   422            X:=X+WLENGTH;
3505   1   29:4   427            PTR:=PTR+WLENGTH;
3506   1   29:3   432          UNTIL DONE;
3507   1   29:3   435          READJUST(PARAPTR,(BUFSIZE-CURSOR+PTR+1)-BUFCOUNT);
3508   1   29:3   447          BUFCOUNT:=BUFSIZE-CURSOR+PTR+1;
3509   1   29:3   456          MOVELEFT(EBUF^[CURSOR],EBUF^[PTR],BUFSIZE-CURSOR+1);
3510   1   29:3   467          EBUF^[BUFCOUNT]:=CHR(0);
3511   1   29:3   471          CURSOR:=MIN(BUFCOUNT-1,SAVE);
3512   1   29:3   481          GETLEADING;
3513   1   29:3   483          CURSOR:=MAX(CURSOR,STUFFSTART)
3514   1   29:2   485      END;
3515   1   29:0   491  END;
3516   1   29:0   512
3517   1   30:D     1  PROCEDURE GETNAME(*MSG:STRING; VAR M:NAME*);
3518   1   30:D    44  VAR
3519   1   30:D    44    I: INTEGER;
3520   1   30:D    45    S: STRING;
3521   1   30:0     0  BEGIN
3522   1   30:1     0    NEEDPROMPT:=TRUE; HOME; CLEARLINE(0); WRITE(MSG,' WHAT MARKER? ');
3523   1   30:1    47    READLN(S);
3524   1   30:1    62    FOR I:=1 TO LENGTH(S) DO S[I]:=UCLC(S[I]);
3525   1   30:1   100    MOVELEFT(S[1],M[0],MIN(8,LENGTH(S)));
3526   1   30:1   116    FILLCHAR(M[LENGTH(S)],MAX(0,8-LENGTH(S)),' ')
3527   1   30:0   135  END;
```

```
3528  1  30:0  150
3529  1  30:0  150
3530  1  36:0    1 PROCEDURE DISKERR;
3531  1  36:0    0 BEGIN
3532  1  36:1    0   ERROR('BAD DISK TRANSFER.',NONFATAL);
3533  1  36:0   24 END;
3534  1  36:0   36
3535  1  34:0    3 FUNCTION WRITEIT(*WHICH:LEFTRIGHT):BOOLEAN*);
3536  1  34:0    4 VAR
3537  1  34:0    4   FULL: BOOLEAN;
3538  1  34:0    0 BEGIN
3539  1  34:1    0   FULL:=(LPAGE+1>=RPAGE);
3540  1  34:1   11   IF NOT FULL THEN
3541  1  34:2   15     BEGIN
3542  1  34:3   15       IF WHICH=LEFTSTACK THEN
3543  1  34:4   20         BEGIN
3544  1  34:5   20           LPAGE:=LPAGE+1;
3545  1  34:5   28           IF BLOCKWRITE(THEFILE,PAGEBUFFER,2,LPAGE+LPAGE)<>2 THEN DISKERR
3546  1  34:4   53         END
3547  1  34:3   55       ELSE
3548  1  34:4   57         BEGIN
3549  1  34:5   57           RPAGE:=RPAGE-1;
3550  1  34:5   65           IF BLOCKWRITE(THEFILE,PAGEBUFFER,2,RPAGE+RPAGE)<>2 THEN DISKERR
3551  1  34:4   90         END
3552  1  34:2   92     END;
3553  1  34:1   92   WRITEIT:=NOT FULL
3554  1  34:0   92 END;
3555  1  34:0  108
3556  1  33:0    3 FUNCTION READIT(*WHICH:LEFTRIGHT): BOOLEAN*);
3557  1  33:0    4 VAR
3558  1  33:0    4   TAPCITY: BOOLEAN;
3559  1  33:0    0 BEGIN
3560  1  33:1    0   TAPCITY:=((WHICH=LEFTSTACK) AND (LPAGE<=0)) OR
3561  1  33:1    9             ((WHICH=RIGHTSTACK) AND (RPAGE>=FLENGTH));
3562  1  33:1   23   IF NOT TAPCITY THEN
3563  1  33:2   27     BEGIN
3564  1  33:3   27       IF WHICH=LEFTSTACK THEN
3565  1  33:4   32         BEGIN
3566  1  33:5   32           IF BLOCKREAD(THEFILE,PAGEBUFFER,2,LPAGE+LPAGE)<>2 THEN DISKERR;
3567  1  33:5   59           LPAGE:=LPAGE-1
3568  1  33:4   62         END
```

```
3569  1  33:3   67        ELSE
3570  1  33:4   69          BEGIN
3571  1  33:5   69            IF BLOCKREAD(THEFILE,PAGEBUFFER,2,RPAGE+RPAGE)<>2 THEN DISKERR;
3572  1  33:5   96            RPAGE:=RPAGE+1
3573  1  33:4   99          END
3574  1  33:2  104      END;
3575  1  33:1  104    READIT:=NOT TAPCITY;
3576  1  33:   108  END;
3577  1  33:0  120
3578  1  31:D    1  PROCEDURE GETPAGES(*WHICH:LEFTRIGHT*);
3579  1  31:D    2  (*WHICH IS WHICH STACK YOU WANT TO READ FROM.  STOPPING CONDITION: APPROXIMATELY
3580  1  31:D    2    2000 CHARACTERS OF SLOP LEFT IN THE BUFFER OR NO MORE STUFF TO READ *)
3581  1  31:D    2  VAR
3582  1  31:D    2    I,START,STUFFCOUNT,THEREST,NOTNULLS: INTEGER;
3583  1  31:D    7    NOTDONE: BOOLEAN;
3584  1  31:0    0  BEGIN
3585  1  31:1    0    IF COPYSTART>BUFCOUNT THEN COPYOK:=FALSE; (* TRASH COPY BUFFER *)
3586  1  31:1   11    NOTDONE:=TRUE;
3587  1  31:1   14    IF WHICH=RIGHTSTACK THEN
3588  1  31:2   19      BEGIN
3589  1  31:3   19        START:=BUFCOUNT;
3590  1  31:3   22        WHILE (START<BUFSIZE-3000) AND NOTDONE DO
3591  1  31:4   33          BEGIN
3592  1  31:5   33            NOTDONE:=READIT(WHICH);
3593  1  31:5   40            IF NOTDONE THEN
3594  1  31:6   43              BEGIN
3595  1  31:7   43                NOTNULLS:=SCAN(-1024,<>CHR(0),PAGEBUFFER[1023])+1024;
3596  1  31:7   64                MOVELEFT(PAGEBUFFER,EBUF^[START],NOTNULLS);
3597  1  31:7   73                WITH PAGEZERO DO (* SWAP IN MARKERS *)
3598  1  31:8   73                  FOR I:=0 TO COUNT-1 DO
3599  1  31:9   88                    IF PAGEN[I]=RPAGE-1 THEN
3600  1  31:0  103                      BEGIN
3601  1  31:1  103                        PAGEN[I]:=-1;
3602  1  31:1  112                        POFFSET[I]:=POFFSET[I]+START;
3603  1  31:0  128                      END;
3604  1  31:7  135                START:=START+NOTNULLS;
3605  1  31:7  140                WRITE('.')
3606  1  31:6  148              END
3607  1  31:4  148          END;
3608  1  31:3  150        BUFCOUNT:=START;
3609  1  31:3  153        EBUF^[BUFCOUNT]:=CHR(0);
```

```
3610  1  31:2  157      END
3611  1  31:1  157    ELSE
3612  1  31:2  159      BEGIN (* LEFTSTACK *)
3613  1  31:3  159        THEREST:=BUFSIZE-BUFCOUNT+1;
3614  1  31:3  166        START:=THEREST-1;
3615  1  31:3  171        READJUST(1,START);
3616  1  31:3  175        MOVERIGHT(EBUF^[1],EBUF^[THEREST],BUFCOUNT);
3617  1  31:3  182        WHILE (START>=3000) AND NOTDONE DO
3618  1  31:4  191          BEGIN
3619  1  31:5  191            NOTDONE:=READIT(WHICH);
3620  1  31:5  198            IF NOTDONE THEN
3621  1  31:6  201              BEGIN
3622  1  31:7  201                NOTNULLS:=SCAN(-MAXCHAR,<>CHR(0),PAGEBUFFER[1023])+1024;
3623  1  31:7  222                MOVELEFT(PAGEBUFFER,EBUF^[START-NOTNULLS+1],NOTNULLS);
3624  1  31:7  235                START:=START-NOTNULLS;
3625  1  31:7  240                WITH PAGEZERO DO (* SWAP IN MARKERS *)
3626  1  31:8  240                  FOR I:=0 TO COUNT-1 DO
3627  1  31:9  255                    IF PAGEN[I]=LPAGE+1 THEN
3628  1  31:0  270                      BEGIN
3629  1  31:1  270                        PAGEN[I]:=-1;
3630  1  31:1  279                        POFFSET[I]:=POFFSET[I]+START+1;
3631  1  31:0  297                      END;
3632  1  31:7  304                WRITE('.');
3633  1  31:6  312              END
3634  1  31:4  312          END;
3635  1  31:3  314        STUFFCOUNT:=BUFSIZE-START;
3636  1  31:3  319        CURSOR:=CURSOR+STUFFCOUNT-BUFCOUNT;
3637  1  31:3  326        READJUST(1,-START);
3638  1  31:3  331        BUFCOUNT:=STUFFCOUNT;
3639  1  31:3  334        MOVELEFT(EBUF^[START+1],EBUF^[1],STUFFCOUNT);
3640  1  31:2  343      END;
3641  1  31:1  343    EBUF^[BUFCOUNT]:=CHR(0);
3642  1  31:0  347  END;
3643  1  31:0  372
3644  1  32:0    1  PROCEDURE PUTPAGES(*WHICH:LEFTRIGHT*);
3645  1  32:0    2  (* IF WHICH=LEFTSTACK THEN SWAP OUT TO THE LEFT STACK OTHERWISE SWAP OUT TO THE
3646  1  32:0    2     RIGHT STACK.  *)
3647  1  32:0    2  VAR
3648  1  32:0    2    I,STOPMARK,SAVE,ONEPAGE,PTR,LAST: INTEGER;
3649  1  32:0    8    OK: BOOLEAN;
3650  1  32:0    9
```

```
3651  1  37:0   3 FUNCTION MOVEITOUT(START,STOP:INTEGER): BOOLEAN;
3652  1  37:0   5 VAR I: INTEGER;
3653  1  37:0   0 BEGIN
3654  1  37:1   0   IF STOP>=START THEN
3655  1  37:2   5     BEGIN
3656  1  37:3   5       MOVELEFT(EBUF^[START],PAGEBUFFER,STOP-START+1);
3657  1  37:3  18       FILLCHAR(PAGEBUFFER[STOP-START+1],1023-(STOP-START),CHR(0));
3658  1  37:3  36       MOVEITOUT:=WRITEIT(WHICH);
3659  1  37:3  45       WITH PAGEZERO DO (* SWAP OUT MARKERS *)
3660  1  37:4  45         FOR I:=0 TO COUNT-1 DO
3661  1  37:5  60           IF (PAGEN[I]=-1) AND (POFFSET[I]>=START) AND (POFFSET[I]<=STOP) THEN
3662  1  37:6  92             BEGIN
3663  1  37:7  92               IF WHICH=LEFTSTACK THEN PAGEN[I]:=LPAGE
3664  1  37:7 105               ELSE PAGEN[I]:=RPAGE;
3665  1  37:7 121               POFFSET[I]:=POFFSET[I]-START;
3666  1  37:6 137             END;
3667  1  37:3 144       WRITE('.')
3668  1  37:2 152     END
3669  1  37:1 152   ELSE
3670  1  37:2 154     MOVEITOUT:=FALSE
3671  1  37:0 154 END;
3672  1  37:0 174
3673  1  32:0   0 BEGIN (* PUTPAGES *)
3674  1  32:1   0   IF WHICH=LEFTSTACK THEN
3675  1  32:2   5     BEGIN
3676  1  32:3   5       LAST:=MAX(CURSOR-200,1); (* SLOP IS 200 *)
3677  1  32:3  17       PTR:=1;
3678  1  32:3  20       REPEAT
3679  1  32:4  20         ONEPAGE:=MIN(PTR+1022,LAST);
3680  1  32:4  32         STOPMARK:=SCAN(-MAXCHAR,=CHR(EOL),EBUF^[ONEPAGE])+ONEPAGE;
3681  1  32:4  47         IF PTR < STOPMARK THEN
3682  1  32:5  52           BEGIN
3683  1  32:6  52             OK:=MOVEITOUT(PTR,STOPMARK);
3684  1  32:6  60             IF OK THEN
3685  1  32:7  63               PTR:=STOPMARK+1
3686  1  32:6  64             ELSE
3687  1  32:7  70               ERROR('RAN OUT OF DISK ROOM',NONFATAL);
3688  1  32:5  96           END
3689  1  32:4  96         ELSE
3690  1  32:5  98           OK:=FALSE;
3691  1  32:3 101       UNTIL NOT OK OR (ONEPAGE  AST);
```

```
3692  1  32:3  109        (* PTR NOW POINTS TO THE FIRST VALID CHARACTER IN THE BUFFER *)
3693  1  32:3  109        IF COPYSTART<PTR THEN COPYOK:=FALSE
3694  1  32:3  116        ELSE
3695  1  32:4  122          IF COPYOK AND (COPYSTART<BUFCOUNT) THEN COPYSTART:=COPYSTART-PTR+1;
3696  1  32:3  143        BUFCOUNT:=BUFCOUNT-PTR+1;
3697  1  32:3  150        MOVELEFT(EBUF^[PTR],EBUF^[1],BUFCOUNT-1);
3698  1  32:3  159        EBUF^[BUFCOUNT]:=CHR(0);
3699  1  32:3  163        (* NOW SHIFT OVER THE MARKERS THAT ARE STILL IN -
3700  1  32:3  163           NOTE THAT READJUST CAN'T BE USED HERE BECAUSE THE
3701  1  32:3  163           MARKERS WANT TO GET SHIFTED PAST PTR *)
3702  1  32:3  163        WITH PAGEZERO DO
3703  1  32:4  163          FOR I:=0 TO COUNT-1 DO
3704  1  32:5  178            IF PAGEN[I]=-1 THEN
3705  1  32:6  190              BEGIN
3706  1  32:7  190                POFFSET[I]:=MAX(1,POFFSET[I]-PTR+1);
3707  1  32:6  213              END;
3708  1  32:3  220        CURSOR:=CURSOR-PTR+1;
3709  1  32:2  227      END
3710  1  32:1  227    ELSE
3711  1  32:2  229      BEGIN (* RIGHT *)
3712  1  32:3  229        PTR:=BUFCOUNT-1;
3713  1  32:3  234        SAVE:=CURSOR;
3714  1  32:3  237        CURSOR:=MIN(CURSOR+200,BUFCOUNT-1);
3715  1  32:3  251        GETLEADING;
3716  1  32:3  253        LAST:=LINESTART;
3717  1  32:3  256        REPEAT
3718  1  32:4  256          ONEPAGE:=MAX(PTR-1022,LAST);
3719  1  32:4  268          IF ONEPAGE=LAST THEN
3720  1  32:5  273            STOPMARK:=ONEPAGE
3721  1  32:4  273          ELSE
3722  1  32:5  278            STOPMARK:=SCAN(MAXCHAR,=CHR(EOL),EBUF^[ONEPAGE])+ONEPAGE+1;
3723  1  32:4  294          IF STOPMARK < PTR THEN
3724  1  32:5  299            BEGIN
3725  1  32:6  299              OK:=MOVEITOUT(STOPMARK,PTR);
3726  1  32:6  307              IF OK THEN
3727  1  32:7  310                PTR:=STOPMARK-1
3728  1  32:6  311              ELSE
3729  1  32:7  317                ERROR('RAN OUT OF DISK ROOM',NONFATAL);
3730  1  32:5  343            END
3731  1  32:4  343          ELSE
3732  1  32:5  345            OK:=FALSE;
```

```
3733   1    32:3    348        UNTIL (ONEPAGE=LAST) OR NOT OK;
3734   1    32:3    356        COPYOK:=(COPYOK AND (COPYSTART>BUFCOUNT)) OR
3735   1    32:3    365                (COPYOK AND (COPYSTART+COPYLENGTH<LAST));
3736   1    32:3    382        BUFCOUNT:=LAST;
3737   1    32:3    385        EBUF^[BUFCOUNT]:=CHR(0);
3738   1    32:3    389        CURSOR:=MIN(BUFCOUNT-1,SAVE)
3739   1    32:2    393      END
3740   1    32:0    399 END;
3741   1    32:0    422
3742   1    35:0      1 PROCEDURE CHECKINDENT(*VAR CURSOR:PTRTYPE*);
3743   1    35:0      2 (* CHECK TO MAKE SURE THAT THE LINE POINTED TO BY CURSOR HAS A LEGITIMATE
3744   1    35:0      2    (ACCORDING TO THE COMPILER, I.E. ONLY ONE DLE) INDENTATION PART.  IF NOT
3745   1    35:0      2    THEN MAKE IT SO *)
3746   1    35:0      0 BEGIN
3747   1    35:1      0   GETLEADING;
3748   1    35:1      2   IF STUFFSTART-LINESTART>2 THEN (* POTENTIALLY TROUBLE! *)
3749   1    35:2      9     BEGIN
3750   1    35:3      9       MOVELEFT(EBUF^[STUFFSTART],EBUF^[LINESTART+2],BUFCOUNT-STUFFSTART);
3751   1    35:3     20       READJUST(LINESTART,LINESTART+2-STUFFSTART);
3752   1    35:3     28       CURSOR:=CURSOR+LINESTART+2-STUFFSTART;
3753   1    35:3     38       BUFCOUNT:=BUFCOUNT+LINESTART+2-STUFFSTART;
3754   1    35:3     47       EBUF^[LINESTART]:=CHR(DLE);
3755   1    35:3     51       EBUF^[LINESTART+1]:=CHR(BLANKS+32)
3756   1    35:2     58     END;
3757   1    35:0     59 END;
3758   1    35:0     72
3759   1    35:0     72 (*$TE D I T O R*)
3760   1    35:0     72
3761   1     1:0      0 BEGIN (* SEGMENT PROCEDURE EDITOR *)
3762   1     1:1      0   INITIALIZE; GETLEADING; CURSOR:=MAX(CURSOR,STUFFSTART);
3763   1     1:1     39   REPEAT
3764   1     1:2     39     CENTERCURSOR(TRASH,(SCREENHEIGHT DIV 2)+1,TRUE);
3765   1     1:2     50     NEEDPROMPT:=TRUE;
3766   1     1:2     54     IF USERINFO.ERRBLK>0 THEN PUTSYNTAX;
3767   1     1:2     64     REPEAT
3768   1     1:3     64       HOME; CLEARLINE(0);
3769   1     1:3     69       EDITCORE;
3770   1     1:3     72       IF COMMAND=SETC THEN ENVIRONMENT
3771   1     1:3     77       ELSE IF COMMAND=COPYC THEN COPYFILE
3772   1     1:2     87     UNTIL COMMAND=QUITC;
3773   1     1:1     95   UNTIL OUT;
```

```
3774   1     1:1   102   SYSCOM^.MISCINFO.NOBREAK := FALSE  (* 28 SEPT 77*)
3775   1     1:0   109 END;
3776   1     1:0   142
3777   0     1:0     0 BEGIN END.
```

```
 1    1     1:0      1 (*$L PRINTER:*)
 1    1     1:0      1 (*$I GLOBALS.TEXT*)
 2    1     1:0      1 (*$U-*)
 3    1     1:0      1 (*$S+*)
 4    1     1:0      1     (*************************************************************)
 5    1     1:0      1     (*                                                           *)
 6    1     1:0      1     (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
 7    1     1:0      1     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-    *)
 8    1     1:0      1     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE   *)
 9    1     1:0      1     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
10    1     1:0      1     (*                                                           *)
11    1     1:0      1     (*************************************************************)
12    1     1:0      1
13    0     1:0      1 PROGRAM PASCALSYSTEM;
14    0     1:0      1
15    0     1:0      1 (*************************************************)
16    0     1:0      1 (*                                               *)
17    0     1:0      1 (*    UCSD PASCAL OPERATING SYSTEM               *)
18    0     1:0      1 (*                                               *)
19    0     1:0      1 (*    RELEASE LEVEL:  I.3   AUGUST, 1977         *)
20    0     1:0      1 (*                    I.4   JANUARY, 1978        *)
21    0     1:0      1 (*                    I.5   SEPTEMBER, 1978      *)
22    0     1:0      1 (*                                               *)
23    0     1:0      1 (*    WRITTEN BY ROGER T. SUMNER                 *)
24    0     1:0      1 (*    WINTER 1977                                *)
25    0     1:0      1 (*                                               *)
26    0     1:0      1 (*    INSTITUTE FOR INFORMATION SYSTEMS          *)
27    0     1:0      1 (*    UC SAN DIEGO, LA JOLLA, CA                 *)
28    0     1:0      1 (*                                               *)
29    0     1:0      1 (*    KENNETH L. BOWLES, DIRECTOR                *)
30    0     1:0      1 (*                                               *)
31    0     1:0      1 (*************************************************)
32    0     1:0      1
33    0     1:0      1 CONST
34    0     1:0      1       MMAXINT = 32767;    (*MAXIMUM INTEGER VALUE*)
35    0     1:0      1       MAXUNIT = 12;       (*MAXIMUM PHYSICAL UNIT # FOR UREAD*)
36    0     1:0      1       MAXDIR = 77;        (*MAX NUMBER OF ENTRIES IN A DIRECTORY*)
37    0     1:0      1       VIDLENG = 7;        (*NUMBER OF CHARS IN A VOLUME ID*)
38    0     1:0      1       TIDLENG = 15;       (*NUMBER OF CHARS IN TITLE ID*)
39    0     1:0      1       MAXSEG = 15;        (*MAX CODE SEGMENT NUMBER*)
40    0     1:0      1       FBLKSIZE = 512;     (*STANDARD DISK BLOCK LENGTH*)
```

```
41   0    1:0     1      DIRBLK = 2;          (*DISK ADDR OF DIRECTORY*)
42   0    1:0     1      AGELIMIT = 300;      (*MAX AGE FOR GDIRP...IN TICKS*)
43   0    1:0     1      EOL = 13;            (*END-OF-LINE...ASCII CR*)
44   0    1:0     1      DLE = 16;            (*BLANK COMPRESSION CODE*)
45   0    1:0     1      NAME_LEN = 23;       [LENGTH OF CONCAT(VIDLENG,':',TIDLENG)]
46   0    1:0     1      FILL_LEN = 11;       [MAXIMUM # OF NULLS IN FILLER]
47   0    1:0     1
48   0    1:0     1  TYPE
49   0    1:0     1
50   0    1:0     1      IORSLTWD = (INOERROR,IBADBLOCK,IBADUNIT,IBADMODE,ITIMEOUT,
51   0    1:0     1                  ILOSTUNIT,ILOSTFILE,IBADTITLE,INOROOM,INOUNIT,
52   0    1:0     1                  INOFILE,IDUPFILE,INOTCLOSED,INOTOPEN,IBADFORMAT,
53   0    1:0     1                  ISTRGOVFL);
54   0    1:0     1
55   0    1:0     1                                        (*COMMAND STATES...SEE GETCMD*)
56   0    1:0     1
57   0    1:0     1      CMDSTATE = (HALTINIT,DEBUGCALL,
58   0    1:0     1                  UPROGNOU,UPROGUOK,SYSPROG,
59   0    1:0     1                  COMPONLY,COMPANDGO,COMPDEBUG,
60   0    1:0     1                  LINKANDGO,LINKDEBUG);
61   0    1:0     1
62   0    1:0     1                                        (*CODE FILES USED IN GETCMD*)
63   0    1:0     1
64   0    1:0     1      SYSFILE = (ASSMBLER,COMPILER,EDITOR,FILER,LINKER);
65   0    1:0     1
66   0    1:0     1                                        (*ARCHIVAL INFO...THE DATE*)
67   0    1:0     1
68   0    1:0     1      DATEREC = PACKED RECORD
69   0    1:0     1                  MONTH: 0..12;        (*0 IMPLIES DATE NOT MEANINGFUL*)
70   0    1:0     1                  DAY: 0..31;          (*DAY OF MONTH*)
71   0    1:0     1                  YEAR: 0..100         (*100 IS TEMP DISK FLAG*)
72   0    1:0     1                END (*DATEREC*) ;
73   0    1:0     1
74   0    1:0     1                                        (*VOLUME TABLES*)
75   0    1:0     1      UNITNUM = 0..MAXUNIT;
76   0    1:0     1      VID = STRING[VIDLENG];
77   0    1:0     1
78   0    1:0     1                                        (*DISK DIRECTORIES*)
79   0    1:0     1      DIRRANGE = 0..MAXDIR;
80   0    1:0     1      TID = STRING[TIDLENG];
81   0    1:0     1      FULL_ID = STRING[NAME_LEN];
```

```
 82   0    1:0     1
 83   0    1:0     1      FILE_TABLE = ARRAY [SYSFILE] OF FULL_ID;
 84   0    1:0     1
 85   0    1:0     1      FILEKIND = (UNTYPEDFILE,XDSKFILE,CODEFILE,TEXTFILE,
 86   0    1:0     1                  INFOFILE,DATAFILE,GRAFFILE,FOTOFILE,SECUREDIR);
 87   0    1:0     1
 88   0    1:0     1      DIRENTRY = PACKED RECORD
 89   0    1:0     1                   DFIRSTBLK: INTEGER;    (*FIRST PHYSICAL DISK ADDR*)
 90   0    1:0     1                   DLASTBLK: INTEGER;     (*POINTS AT BLOCK FOLLOWING*)
 91   0    1:0     1                   CASE DFKIND: FILEKIND OF
 92   0    1:0     1                     SECUREDIR,
 93   0    1:0     1                     UNTYPEDFILE: (*ONLY IN DIR[0]...VOLUME INFO*)
 94   0    1:0     1                        (FILLER1 : 0..2048; [FOR DOWNWARD COMPATIBILITY,13 BITS]
 95   0    1:0     1                         DVID: VID;                (*NAME OF DISK VOLUME*)
 96   0    1:0     1                         DEOVBLK: INTEGER;         (*LASTBLK OF VOLUME*)
 97   0    1:0     1                         DNUMFILES: DIRRANGE;      (*NUM FILES IN DIR*)
 98   0    1:0     1                         DLOADTIME: INTEGER;       (*TIME OF LAST ACCESS*)
 99   0    1:0     1                         DLASTBOOT: DATEREC);      (*MOST RECENT DATE SETTING*)
100   0    1:0     1                     XDSKFILE,CODEFILE,TEXTFILE,INFOFILE,
101   0    1:0     1                     DATAFILE,GRAFFILE,FOTOFILE:
102   0    1:0     1                        (FILLER2 : 0..512; [FOR DOWNWARD COMPATIBILITY]
103   0    1:0     1                         STATUS : BOOLEAN;          [FOR FILER WILDCARDS]
104   0    1:0     1                         DTID: TID;                (*TITLE OF FILE*)
105   0    1:0     1                         DLASTBYTE: 1..FBLKSIZE;   (*NUM BYTES IN LAST BLOCK*)
106   0    1:0     1                         DACCESS: DATEREC)         (*LAST MODIFICATION DATE*)
107   0    1:0     1                END (*DIRENTRY*) ;
108   0    1:0     1
109   0    1:0     1      DIRP = ^DIRECTORY;
110   0    1:0     1
111   0    1:0     1      DIRECTORY = ARRAY [DIRRANGE] OF DIRENTRY;
112   0    1:0     1
113   0    1:0     1                                            (*FILE INFORMATION*)
114   0    1:0     1
115   0    1:0     1      CLOSETYPE = (CNORMAL,CLOCK,CPURGE,CCRUNCH);
116   0    1:0     1      WINDOWP = ^WINDOW;
117   0    1:0     1      WINDOW = PACKED ARRAY [0..0] OF CHAR;
118   0    1:0     1      FIBP = ^FIB;
119   0    1:0     1
120   0    1:0     1      FIB = RECORD
121   0    1:0     1              FWINDOW: WINDOWP;  (*USER WINDOW...F^, USED BY GET-PUT*)
122   0    1:0     1              FEOF,FEOLN: BOOLEAN;
```

```
123   0    1:0     1              FSTATE: (FJANDW,FNEEDCHAR,FGOTCHAR);
124   0    1:0     1              FRECSIZE: INTEGER; (*IN BYTES...0=>BLOCKFILE, 1=>CHARFILE*)
125   0    1:0     1              CASE FISOPEN: BOOLEAN OF
126   0    1:0     1                TRUE: (FISBLKD: BOOLEAN; (*FILE IS ON BLOCK DEVICE*)
127   0    1:0     1                       FUNIT: UNITNUM;   (*PHYSICAL UNIT #*)
128   0    1:0     1                       FVID: VID;        (*VOLUME NAME*)
129   0    1:0     1                       FREPTCNT,         (* # TIMES F^ VALID W/O GET*)
130   0    1:0     1                       FNXTBLK,          (*NEXT REL BLOCK TO IO*)
131   0    1:0     1                       FMAXBLK: INTEGER; (*MAX REL BLOCK ACCESSED*)
132   0    1:0     1                       FMODIFIED:BOOLEAN;(*PLEASE SET NEW DATE IN CLOSE*)
133   0    1:0     1                       FHEADER: DIRENTRY;(*COPY OF DISK DIR ENTRY*)
134   0    1:0     1                       CASE FSOFTBUF: BOOLEAN OF (*DISK GET-PUT STUFF*)
135   0    1:0     1                         TRUE: (FNXTBYTE,FMAXBYTE: INTEGER;
136   0    1:0     1                                FBUFCHNGD: BOOLEAN;
137   0    1:0     1                                FBUFFER: PACKED ARRAY [0..FBLKSIZE] OF CHAR))
138   0    1:0     1          END (*FIB*) ;
139   0    1:0     1
140   0    1:0     1                                          (*USER WORKFILE STUFF*)
141   0    1:0     1
142   0    1:0     1      INFOREC = RECORD
143   0    1:0     1                  SYMFIBP,CODEFIBP: FIBP;          (*WORKFILES FOR SCRATCH*)
144   0    1:0     1                  ERRSYM,ERRBLK,ERRNUM: INTEGER;   (*ERROR STUFF IN EDIT*)
145   0    1:0     1                  SLOWTERM,STUPID: BOOLEAN;        (*STUDENT PROGRAMMER ID!!*)
146   0    1:0     1                  ALTMODE: CHAR;                   (*WASHOUT CHAR FOR COMPILER*)
147   0    1:0     1                  GOTSYM,GOTCODE: BOOLEAN;         (*TITLES ARE MEANINGFUL*)
148   0    1:0     1                  WORKVID,SYMVID,CODEVID: VID;     (*PERM&CUR WORKFILE VOLUMES*)
149   0    1:0     1                  WORKTID,SYMTID,CODETID: TID      (*PERM&CUR WORKFILES TITLE*)
150   0    1:0     1                END (*INFOREC*) ;
151   0    1:0     1
152   0    1:0     1                                          (*CODE SEGMENT LAYOUTS*)
153   0    1:0     1
154   0    1:0     1      SEGRANGE = 0..MAXSEG;
155   0    1:0     1      SEGDESC = RECORD
156   0    1:0     1                  DISKADDR: INTEGER;       (*REL BLK IN CODE...ABS IN SYSCOM^*)
157   0    1:0     1                  CODELENG: INTEGER        (*# BYTES TO READ IN*)
158   0    1:0     1                END (*SEGDESC*) ;
159   0    1:0     1
160   0    1:0     1                                          (*DEBUGGER STUFF*)
161   0    1:0     1
162   0    1:0     1      BYTERANGE = 0..255;
163   0    1:0     1      TRICKARRAY = ARRAY [0..0] OF INTEGER; (* FOR MEMORY DIDDLING*)
```

```
164   0    1:0     1      MSCWP = ^ MSCW;              (*MARK STACK RECORD POINTER*)
165   0    1:0     1      MSCW = RECORD
166   0    1:0     1               STATLINK: MSCWP;   (*POINTER TO PARENT MSCW*)
167   0    1:0     1               DYNLINK: MSCWP;    (*POINTER TO CALLER'S MSCW*)
168   0    1:0     1               MSSEG,MSJTAB: ^TRICKARRAY;
169   0    1:0     1               MSIPC: INTEGER;
170   0    1:0     1               LOCALDATA: TRICKARRAY
171   0    1:0     1             END (*MSCW*) ;
172   0    1:0     1
173   0    1:0     1                                          (*SYSTEM COMMUNICATION AREA*)
174   0    1:0     1                                          (*SEE INTERPRETERS...NOTE  *)
175   0    1:0     1                                          (*THAT WE ASSUME BACKWARD  *)
176   0    1:0     1                                          (*FIELD ALLOCATION IS DONE *)
177   0    1:0     1      SYSCOMREC = RECORD
178   0    1:0     1                    IORSLT: IORSLTWD;     (*RESULT OF LAST IO CALL*)
179   0    1:0     1                    XEQERR: INTEGER;      (*REASON FOR EXECERROR CALL*)
180   0    1:0     1                    SYSUNIT: UNITNUM;     (*PHYSICAL UNIT OF BOOTLOAD*)
181   0    1:0     1                    BUGSTATE: INTEGER;    (*DEBUGGER INFO*)
182   0    1:0     1                    GDIRP: DIRP;          (*GLOBAL DIR POINTER,SEE VOLSEARCH*)
183   0    1:0     1                    LASTMP,STKBASE,BOMBP: MSCWP;
184   0    1:0     1                    MEMTOP,SEG,JTAB: INTEGER;
185   0    1:0     1                    BOMBIPC: INTEGER;     (*WHERE XEQERR BLOWUP WAS*)
186   0    1:0     1                    HLTLINE: INTEGER;     (*MORE DEBUGGER STUFF*)
187   0    1:0     1                    BRKPTS: ARRAY [0..3] OF INTEGER;
188   0    1:0     1                    RETRIES: INTEGER;     (*DRIVERS PUT RETRY COUNTS*)
189   0    1:0     1                    EXPANSION: ARRAY [0..8] OF INTEGER;
190   0    1:0     1                    HIGHTIME,LOWTIME: INTEGER;
191   0    1:0     1                    MISCINFO: PACKED RECORD
192   0    1:0     1                                NOBREAK,STUPID,SLOWTERM,
193   0    1:0     1                                HASXYCRT,HASLCCRT,HAS8510A,HASCLOCK: BOOLEAN;
194   0    1:0     1                                USERKIND:(NORMAL, AQUIZ, BOOKER, PQUIZ)
195   0    1:0     1                              END;
196   0    1:0     1                    CRTTYPE: INTEGER;
197   0    1:0     1                    CRTCTRL: PACKED RECORD
198   0    1:0     1                               RLF,NDFS,ERASEEOL,ERASEEOS,HOME,ESCAPE: CHAR;
199   0    1:0     1                               BACKSPACE: CHAR;
200   0    1:0     1                               FILLCOUNT: 0..255;
201   0    1:0     1                               CLEARSCREEN, CLEARLINE: CHAR;
202   0    1:0     1                               PREFIXED: PACKED ARRAY [0..8] OF BOOLEAN
203   0    1:0     1                             END;
204   0    1:0     1                    CRTINFO: PACKED RECORD
```

```
205   0    1:D     1                                WIDTH,HEIGHT: INTEGER;
206   0    1:D     1                                RIGHT,LEFT,DOWN,UP: CHAR;
207   0    1:D     1                                BADCH,CHARDEL,STOP,BREAK,FLUSH,EOF: CHAR;
208   0    1:D     1                                ALTMODE,LINEDEL: CHAR;
209   0    1:D     1                                BACKSPACE,ETX,PREFIX: CHAR;
210   0    1:D     1                                PREFIXED: PACKED ARRAY [0..13] OF BOOLEAN
211   0    1:D     1                              END;
212   0    1:D     1                    SEGTABLE: ARRAY [SEGRANGE] OF
213   0    1:D     1                                RECORD
214   0    1:D     1                                  CODEUNIT: UNITNUM;
215   0    1:D     1                                  CODEDESC: SEGDESC
216   0    1:D     1                                END
217   0    1:D     1                  END (*SYSCOM*);
218   0    1:D     1
219   0    1:D     1      MISCINFOREC = RECORD
220   0    1:D     1                      MSYSCOM: SYSCOMREC
221   0    1:D     1                    END;
222   0    1:D     1
223   0    1:D     1 VAR
224   0    1:D     1     SYSCOM: ^SYSCOMREC;                    (*MAGIC PARAM...SET UP IN BOOT*)
225   0    1:D     2     GFILES: ARRAY [0..5] OF FIBP;          (*GLOBAL FILES, 0=INPUT, 1=OUTPUT*)
226   0    1:D     8     USERINFO: INFOREC;                     (*WORK STUFF FOR COMPILER ETC*)
227   0    1:D    54     EMPTYHEAP: ^INTEGER;                   (*HEAP MARK FOR MEM MANAGING*)
228   0    1:D    55     INPUTFIB,OUTPUTFIB,                    (*CONSOLE FILES...GFILES ARE COPIES*)
229   0    1:D    55     SYSTERM,SWAPFIB: FIBP;                 (*CONTROL AND SWAPSPACE FILES*)
230   0    1:D    59     SYVID,DKVID: VID;                      (*SYSUNIT VOLID & DEFAULT VOLID*)
231   0    1:D    67     THEDATE: DATEREC;                      (*TODAY...SET IN FILER OR SIGN ON*)
232   0    1:D    68     DEBUGINFO: ^INTEGER;                   (*DEBUGGERS GLOBAL INFO WHILE RUNIN*)
233   0    1:D    69     STATE: CMDSTATE;                       (*FOR GETCOMMAND*)
234   0    1:D    70     PL: STRING;                            (*PROMPTLINE STRING...SEE PROMPT*)
235   0    1:D   111     IPOT: ARRAY [0..4] OF INTEGER;         (*INTEGER POWERS OF TEN*)
236   0    1:D   116     FILLER: STRING[FILL_LEN];              (*NULLS FOR CARRIAGE DELAY*)
237   0    1:D   122     DIGITS: SET OF '0'..'9';
238   0    1:D   126     UNITABLE: ARRAY [UNITNUM] OF (*0 NOT USED*)
239   0    1:D   126                 RECORD
240   0    1:D   126                   UVID: VID;    (*VOLUME ID FOR UNIT*)
241   0    1:D   126                   CASE UISBLKD: BOOLEAN OF
242   0    1:D   126                     TRUE: (UEOVBLK: INTEGER)
243   0    1:D   126                 END (*UNITABLE*) ;
244   0    1:D   204     FILENAME : FILE_TABLE;
245   0    1:D   264
```

```
246   0    1:0   264 (*-------------------------------------------------------------------------------*)
247   0    1:0   264 (* SYSTEM PROCEDURE FORWARD DECLARATIONS *)
248   0    1:0   264 (* THESE ARE ADDRESSED BY OBJECT CODE... *)
249   0    1:0   264 (*  DO NOT MOVE WITHOUT CAREFUL THOUGHT  *)
250   0    1:0   264
251   0    2:0     1 PROCEDURE EXECERROR;
252   0    2:0     1   FORWARD;
253   0    3:0     1 PROCEDURE FINIT(VAR F: FIB; WINDOW: WINDOWP; RECWORDS: INTEGER);
254   0    3:0     4   FORWARD;
255   0    4:0     1 PROCEDURE FRESET(VAR F: FIB);
256   0    4:0     2   FORWARD;
257   0    5:0     1 PROCEDURE FOPEN(VAR F: FIB; VAR FTITLE: STRING;
258   0    5:0     3                     FOPENOLD: BOOLEAN; JUNK: FIBP);
259   0    5:0     5   FORWARD;
260   0    6:0     1 PROCEDURE FCLOSE(VAR F: FIB; FTYPE: CLOSETYPE);
261   0    6:0     3   FORWARD;
262   0    7:0     1 PROCEDURE FGET(VAR F: FIB);
263   0    7:0     2   FORWARD;
264   0    8:0     1 PROCEDURE FPUT(VAR F: FIB);
265   0    8:0     2   FORWARD;
266   0    9:0     1 PROCEDURE XSEEK;
267   0    9:0     1   FORWARD;
268   0   10:0     3 FUNCTION FEOF(VAR F: FIB): BOOLEAN;
269   0   10:0     4   FORWARD;
270   0   11:0     3 FUNCTION FEOLN(VAR F: FIB): BOOLEAN;
271   0   11:0     4   FORWARD;
272   0   12:0     1 PROCEDURE FREADINT(VAR F: FIB; VAR I: INTEGER);
273   0   12:0     3   FORWARD;
274   0   13:0     1 PROCEDURE FWRITEINT(VAR F: FIB; I,RLENG: INTEGER);
275   0   13:0     4   FORWARD;
276   0   14:0     1 PROCEDURE XREADREAL;
277   0   14:0     1   FORWARD;
278   0   15:0     1 PROCEDURE XWRITEREAL;
279   0   15:0     1   FORWARD;
280   0   16:0     1 PROCEDURE FREADCHAR(VAR F: FIB; VAR CH: CHAR);
281   0   16:0     3   FORWARD;
282   0   17:0     1 PROCEDURE FWRITECHAR(VAR F: FIB; CH: CHAR; RLENG: INTEGER);
283   0   17:0     4   FORWARD;
284   0   18:0     1 PROCEDURE FREADSTRING(VAR F: FIB; VAR S: STRING; SLENG: INTEGER);
285   0   18:0     4   FORWARD;
286   0   19:0     1 PROCEDURE FWRITESTRING(VAR F: FIB; VAR S: STRING; RLENG: INTEGER);
```

```
287   0   19:D     4   FORWARD;
288   0   20:D     1 PROCEDURE FWRITEBYTES(VAR F: FIB; VAR A: WINDOW; RLENG,ALENG: INTEGER);
289   0   20:D     5   FORWARD;
290   0   21:D     1 PROCEDURE FREADLN(VAR F: FIB);
291   0   21:D     2   FORWARD;
292   0   22:D     1 PROCEDURE FWRITELN(VAR F: FIB);
293   0   22:D     2   FORWARD;
294   0   23:D     1 PROCEDURE SCONCAT(VAR DEST,SRC: STRING; DESTLENG: INTEGER);
295   0   23:D     4   FORWARD;
296   0   24:D     1 PROCEDURE SINSERT(VAR SRC,DEST: STRING; DESTLENG,INSINX: INTEGER);
297   0   24:D     5   FORWARD;
298   0   25:D     1 PROCEDURE SCOPY(VAR SRC,DEST: STRING; SRCINX,COPYLENG: INTEGER);
299   0   25:D     5   FORWARD;
300   0   26:D     1 PROCEDURE SDELETE(VAR DEST: STRING; DELINX,DELLENG: INTEGER);
301   0   26:D     4   FORWARD;
302   0   27:D     3 FUNCTION SPOS(VAR TARGET,SRC: STRING): INTEGER;
303   0   27:D     5   FORWARD;
304   0   28:D     3 FUNCTION FBLOCKIO(VAR F: FIB; VAR A: WINDOW; I: INTEGER;
305   0   28:D     6                   NBLOCKS,RBLOCK: INTEGER; DOREAD: BOOLEAN): INTEGER;
306   0   28:D     9   FORWARD;
307   0   29:D     1 PROCEDURE FGOTOXY(X,Y: INTEGER);
308   0   29:D     3   FORWARD;
309   0   29:D     3
310   0   29:D     3 (* NON FIXED FORWARD DECLARATIONS *)
311   0   29:D     3
312   0   30:D     3 FUNCTION VOLSEARCH(VAR FVID: VID; LOOKHARD: BOOLEAN;
313   0   30:D     5                    VAR FDIR: DIRP): UNITNUM;
314   0   30:D     6   FORWARD;
315   0   31:D     1 PROCEDURE WRITEDIR(FUNIT: UNITNUM; FDIR: DIRP);
316   0   31:D     3   FORWARD;
317   0   32:D     3 FUNCTION DIRSEARCH(VAR FTID: TID; FINDPERM: BOOLEAN; FDIR: DIRP): DIRRANGE;
318   0   32:D     6   FORWARD;
319   0   33:D     3 FUNCTION SCANTITLE(FTITLE: STRING; VAR FVID: VID; VAR FTID: TID;
320   0   33:D     6                    VAR FSEGS: INTEGER; VAR FKIND: FILEKIND): BOOLEAN;
321   0   33:D    49   FORWARD;
322   0   34:D     1 PROCEDURE DELENTRY(FINX: DIRRANGE; FDIR: DIRP);
323   0   34:D     3   FORWARD;
324   0   35:D     1 PROCEDURE INSENTRY(VAR FENTRY: DIRENTRY; FINX: DIRRANGE; FDIR: DIRP);
325   0   35:D     4   FORWARD;
326   0   36:D     1 PROCEDURE HOMECURSOR;
327   0   36:D     1   FORWARD;
```

```
328   0    37:D     1 PROCEDURE CLEARSCREEN;
329   0    37:D     1   FORWARD;
330   0    38:D     1 PROCEDURE CLEARLINE;
331   0    38:D     1   FORWARD;
332   0    39:D     1 PROCEDURE PROMPT;
333   0    39:D     1   FORWARD;
334   0    40:D     3 FUNCTION SPACEWAIT(FLUSH: BOOLEAN): BOOLEAN;
335   0    40:D     4   FORWARD;
336   0    41:D     3 FUNCTION GETCHAR(FLUSH: BOOLEAN): CHAR;
337   0    41:D     4   FORWARD;
338   0    42:D     3 FUNCTION FETCHDIR(FUNIT:UNITNUM) : BOOLEAN;
339   0    42:D     4   FORWARD;
340   0    43:D     1 PROCEDURE COMMAND;
341   0    43:D     1   FORWARD;
342   0    43:D     1
343   0    43:D     1 (*$I GLOBALS.TEXT*)
344   0    43:D     1
345   1     1:D     1 SEGMENT PROCEDURE YALOE(INN,OWWT: FIBP);
346   1     1:D     3 (*****************************************************************)
347   1     1:D     3 (*   COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.   *)
348   1     1:D     3 (*   PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-    *)
349   1     1:D     3 (*   TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE   *)
350   1     1:D     3 (*   OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
351   1     1:D     3 (*****************************************************************)
352   1     1:D     3
353   1     1:D     3 (* YALOE * YALOE * YALOE * YALOE * YALOE * YALOE * YALOE * YALOE *
354   1     1:D     3  * THIS TEXT EDITOR IS BASED ON THE COMMAND STRUCTURE
355   1     1:D     3  * OF THE RT-11 SYSTEM TEXT EDITOR.  INITIALLY STRUCTURED
356   1     1:D     3  * AND WRITTED BY RICHARD KAUFMANN AND GREG DAVIDSON.
357   1     1:D     3  * LATER MODIFIED, ENHANCED, AND QUICKENED BY KEITH SHILLINGTON.
358   1     1:D     3  * RELEASED CONTINUOUSLY FROM EARLY JUNE 1977.
359   1     1:D     3  * LATEST FIXES BY ROGER SUMNER FOR I.3  8-AUG-77
360   1     1:D     3  * 11-AUG-77 KEITH SHILLINGTON  BACKSPACING CHANGES
361   1     1:D     3  * 13-SEP-77 KAS & RTS ALPHA LOCK AND BACKSPACE FIX
362   1     1:D     3  * 24-SEP-77 RTS REMOVES ALPHA LOCK...PUT INTO 1.3B INTERP
363   1     1:D     3  *  7-OCT-77 MADE A NON-SYSTEM PROGRAM...RSK DYNASTY TAKES OVER
364   1     1:D     3  *  9-FEB-78 BUGS ABOUT HEAP REMAIN...I.4 OUT THE DOOR ANYWAY
365   1     1:D     3  *           SYSTEM WORKS OK WITH DIRTY FIX IN WRITEDIR!
366   1     1:D     3  * YALOE * YALOE * YALOE * YALOE * YALOE * YALOE * YALOE * YALOE *)
367   1     1:D     3
368   1     1:D     3  CONST
```

```
369   1    1:0     3   RET = 13;  TAB = 9;
370   1    1:0     3   CTRLX = (*0300*) 24;
371   1    1:0     3   DC1 =   (*0210*) 17;
372   1    1:0     3   EXECSIZE = 1000;
373   1    1:0     3   MAXMAC = 9; (* CHANGING THIS HAS IMPACT ON THE CODE... *)
374   1    1:0     3   SHIFT = 15;
375   1    1:0     3 TYPE
376   1    1:0     3   FILEBUF = PACKED ARRAY[0..1023] OF CHAR;
377   1    1:0     3   COMARRAY = PACKED ARRAY[0..99] OF CHAR;
378   1    1:0     3   BUFCHUNK = PACKED ARRAY[0..999] OF CHAR;
379   1    1:0     3 VAR
380   1    1:0     3   I,J,ENDPOS,CURSOR: INTEGER;
381   1    1:0     7   BUFSIZE,BUFEND: INTEGER;
382   1    1:0     9   EQUALLENGTH: INTEGER;
383   1    1:0    10   ESC: CHAR;
384   1    1:0    11   CTRLU: INTEGER;
385   1    1:0    12   BACK,ACR: CHAR;
386   1    1:0    14   EXEC: ^COMARRAY;
387   1    1:0    15   BUF: ^BUFCHUNK;
388   1    1:0    16   MACROS: ARRAY[0..MAXMAC] OF
389   1    1:0    16               RECORD
390   1    1:0    16                 LGTH: INTEGER;
391   1    1:0    16                 EXEC: ^COMARRAY
392   1    1:0    16               END;
393   1    1:0    36   OPTION: PACKED RECORD
394   1    1:0    36         LISTSIZE: 0..100;
395   1    1:0    36         ONOFF: BOOLEAN
396   1    1:0    36   END;
397   1    1:0    37   IOFILE: FILE;
398   1    1:0    77
399   1    2:0     3 FUNCTION COMMAND: BOOLEAN; FORWARD;
400   1    2:0     3
401   1    3:0     3 FUNCTION MIN(A,B:INTEGER): INTEGER;
402   1    3:0     0 BEGIN
403   1    3:1     0   IF A>B THEN MIN := B ELSE MIN := A
404   1    3:0    10 END;
405   1    3:0    26
406   1    4:0     3 FUNCTION NEWFIN: BOOLEAN; (* TRUE IF ERROR OCCURS *)
407   1    4:0     3 LABEL 1;
408   1    4:0     3 VAR
409   1    4:0     3   NBLOCKS,STASHSIZE,STASHEDAT: INTEGER;
```

```
410   1    4:0     6     STASHCURSOR,NPAGES,I,NEXT: INTEGER;
411   1    4:0    10     DIDDLED: BOOLEAN;
412   1    4:0     0  BEGIN
413   1    4:1     0    NEWFIN := FALSE;
414   1    4:1     3    IF BLOCKREAD(IOFILE,I,0,2) = 0 THEN BEGIN (* OK *)
415   1    4:3    20      STASHCURSOR := CURSOR;
416   1    4:3    23      STASHSIZE := ENDPOS - CURSOR;  STASHEDAT := BUFEND-STASHSIZE;
417   1    4:3    33      IF (STASHEDAT > CURSOR) THEN (* THERE IS ROOM *)
418   1    4:4    38        MOVERIGHT(BUF^[CURSOR],BUF^[STASHEDAT],STASHSIZE)
419   1    4:3    45      ELSE
420   1    4:4    47        BEGIN
421   1    4:5    47          WRITELN(OUTPUT,'NOT ENOUGH SPACE');
422   1    4:5    79          NEWFIN := TRUE;
423   1    4:5    82          GOTO 1;
424   1    4:4    84        END;
425   1    4:3    84      DIDDLED := FALSE;
426   1    4:3    87      IF ODD(CURSOR) THEN BEGIN
427   1    4:5    90        DIDDLED := TRUE;
428   1    4:5    93        CURSOR := CURSOR +1;
429   1    4:4    98      END;
430   1    4:3    98      NBLOCKS := (STASHEDAT - CURSOR) DIV 512;
431   1    4:3   107      NBLOCKS := BLOCKREAD(IOFILE,BUF^[CURSOR],NBLOCKS);
432   1    4:3   124      IF (NOT EOF(IOFILE)) OR (IORESULT <> 0) OR (ODD(NBLOCKS)) THEN BEGIN
433   1    4:5   141        CLOSE(IOFILE);
434   1    4:5   147        WRITELN(OUTPUT,'NOT ENOUGH SPACE');
435   1    4:5   179        CURSOR := STASHCURSOR;
436   1    4:5   182        NEWFIN := TRUE;
437   1    4:5   185        GOTO 1;
438   1    4:4   187      END;
439   1    4:3   187      NPAGES := NBLOCKS DIV 2;
440   1    4:3   192      IF DIDDLED THEN (* UGH *) BEGIN
441   1    4:5   195        CURSOR := CURSOR -1;
442   1    4:5   200        MOVELEFT(BUF^[CURSOR+1],BUF^[CURSOR],NPAGES*1024);
443   1    4:4   213      END;
444   1    4:3   213      NEXT := CURSOR;
445   1    4:3   216      WHILE NPAGES > 0 DO BEGIN
446   1    4:5   221        NPAGES := NPAGES -1;
447   1    4:5   226        CURSOR := CURSOR +1023;
448   1    4:5   233        NEXT := NEXT +1024;
449   1    4:5   240        I := SCAN(-1024,<>CHR(0),BUF^[CURSOR]);
450   1    4:5   253        CURSOR := CURSOR +I +1; (* POINT AT FIRST NUL *)
```

```
451   1    4:5   260        IF NPAGES > 0 THEN  MOVELEFT(BUF^[NEXT],BUF^[CURSOR],1024);
452   1    4:4   274      END;
453   1    4:3   276 1:       (* THIS IS WHERE THE WOUND IS CLOSED AND HEALED *)
454   1    4:3   276      CLOSE(IOFILE);
455   1    4:3   282      MOVELEFT(BUF^[STASHEDAT],BUF^[CURSOR],STASHSIZE);
456   1    4:3   289      ENDPOS := STASHSIZE +CURSOR;
457   1    4:3   294      BUF^[ENDPOS] := CHR(0);
458   1    4:3   298      CURSOR := STASHCURSOR;
459   1    4:2   301    END;
460   1    4:0   301  END;
461   1    4:0   320
462   1    5:0     1  PROCEDURE INITIALIZE;
463   1    5:0     1  VAR
464   1    5:0     1    BUFMAKER: ^BUFCHUNK;
465   1    5:0     2    SPACEMAKER: ^COMARRAY;
466   1    5:0     3    HERE: ^INTEGER;
467   1    5:0     4    LIMIT: INTEGER;
468   1    5:0     5    TEST: BOOLEAN;
469   1    5:0     0  BEGIN
470   1    5:1     0    WRITE(OUTPUT,'YALOE:');
471   1    5:1    16    IF NOT SYSCOM^.MISCINFO.SLOWTERM THEN
472   1    5:2    27      WRITE(OUTPUT,
473   1    5:2    27          ' - ? <ESC><ESC> FOR DETAILS');
474   1    5:1    64    WRITELN(OUTPUT);
475   1    5:1    70    NEW(BUF); (* BASE OF THE BUFFER *)
476   1    5:1    77    BUFSIZE := SIZEOF(BUFCHUNK);
477   1    5:1    82    LIMIT := ORD(SYSCOM^.LASTMP);
478   1    5:1    88    REPEAT
479   1    5:2    38      MARK(HERE);
480   1    5:2    92      TEST := ((LIMIT - ORD(HERE))<5000) AND ((LIMIT - ORD(HERE))>0);
481   1    5:2   107      IF NOT TEST THEN
482   1    5:3   111        BEGIN
483   1    5:4   111          NEW(BUFMAKER);
484   1    5:4   118          BUFSIZE := BUFSIZE +SIZEOF(BUFCHUNK)
485   1    5:3   122        END;
486   1    5:1   125    UNTIL TEST;
487   1    5:1   128    IF BUFSIZE < 0 THEN BUFSIZE := 32000;
488   1    5:1   138    NEW(EXEC);
489   1    5:1   143    FOR I := 1 TO 9 DO NEW(SPACEMAKER); (* CREATE SPACE FOR BASIC COMMAND *)
490   1    5:1   166    FOR I := 0 TO MAXMAC DO
491   1    5:2   177      MACROS[I].EXEC := NIL;
```

```
492   1     5:1    193    CURSOR := 0;  ENDPOS := 0;
493   1     5:1    199    OPTION.ONOFF := FALSE;
494   1     5:1    205    BUFEND := BUFSIZE;
495   1     5:1    208    I := 0;
496   1     5:1    211    ACR := CHR(RET);
497   1     5:1    214    BACK := SYSCOM^.CRTCTRL.BACKSPACE;
498   1     5:1    224    ESC := SYSCOM^.CRTINFO.ALTMODE;
499   1     5:1    234    CTRLU := ORD(SYSCOM^.CRTINFO.LINEDEL);
500   1     5:1    244    WITH USERINFO DO
501   1     5:2    244     IF GOTSYM THEN BEGIN
502   1     5:4    249      OPENOLD(IOFILE,CONCAT(SYMVID,':',SYMTID));
503   1     5:4    289      IF NEWFIN THEN
504   1     5:5    295      BEGIN WRITELN(OUTPUT,'LOST WORKFILE SOURCE');
505   1     5:6    331        GOTSYM := FALSE
506   1     5:5    331      END
507   1     5:4    335      ELSE BEGIN
508   1     5:6    337        WRITE(OUTPUT,'WORKFILE ');
509   1     5:6    356        IF LENGTH(WORKTID) > 0 THEN
510   1     5:7    365          WRITE(OUTPUT,WORKTID,' ');
511   1     5:6    383        WRITELN(OUTPUT,'READ IN');
512   1     5:5    406       END
513   1     5:3    406     END ELSE BEGIN
514   1     5:4    408      ENDPOS := 0; BUF^[0] := CHR(0);
515   1     5:4    415      WRITELN(OUTPUT,'NO WORKFILE TO READ');
516   1     5:3    450   END;
517   1     5:1    450   CURSOR := 0;
518   1     5:1    453   EQUALLENGTH := 0;
519   1     5:0    456 END;
520   1     5:0    476
521   1     5:0    476
522   1     6:D      1   PROCEDURE NEWOUTLOOK;
523   1     6:D      1   VAR  I:INTEGER;
524   1     6:D      2     STASHCURSOR: INTEGER;
525   1     6:D      3     P: ^INTEGER;
526   1     6:D      4     COM: ^FILEBUF;
527   1     6:0      0   BEGIN
528   1     6:1      0     STASHCURSOR := CURSOR;
529   1     6:1      3     MARK(P);
530   1     6:1      7     NEW(COM);
531   1     6:1     14     FILLCHAR(COM^[0],1024,CHR(0));
532   1     6:1     22     CURSOR := 0;
```

```
533   1    6:1    25      IF BLOCKWRITE(IOFILE,COM^,2) = 2 THEN
534   1    6:2    44      WHILE (CURSOR + 1023) < ENDPOS DO BEGIN
535   1    6:4    53        I := SCAN(-1022, = CHR(RET), BUF^[CURSOR +1022]);
536   1    6:4    70        MOVELEFT(BUF^[CURSOR],COM^,1023+I);
537   1    6:4    81        FILLCHAR(COM^[1023+I],ABS(I)+1,CHR(0));
538   1    6:4    94        IF BLOCKWRITE(IOFILE,COM^,2) <>2 THEN BEGIN
539   1    6:6   113          RELEASE(P);
540   1    6:6   117          WRITELN(OUTPUT, 'OUTPUT FILE ERROR: HELP');
541   1    6:6   156          CLOSE(IOFILE);
542   1    6:6   162          EXIT(COMMAND);
543   1    6:5   166        END;
544   1    6:4   166        CURSOR := CURSOR+1023+I;
545   1    6:3   175      END;
546   1    6:1   177      IF (CURSOR < ENDPOS) THEN BEGIN
547   1    6:3   182        FILLCHAR(BUF^[ENDPOS],1024-(ENDPOS-CURSOR),CHR(0));
548   1    6:3   194        MOVELEFT(BUF^[CURSOR],COM^,1024);
549   1    6:3   203        IF BLOCKWRITE(IOFILE,COM^,2) <>2 THEN BEGIN
550   1    6:5   222          RELEASE(P);
551   1    6:5   226          WRITELN(OUTPUT,'OUTPUT FILE ERROR.  HELP!');
552   1    6:5   267          CLOSE(IOFILE);
553   1    6:5   273          EXIT(COMMAND);
554   1    6:4   277        END;
555   1    6:2   277      END;
556   1    6:1   277      RELEASE(P);
557   1    6:1   281      CLOSE(IOFILE,LOCK);
558   1    6:1   287      CURSOR := STASHCURSOR;
559   1    6:0   290    END;
560   1    6:0   306
561   1    7:D     1  PROCEDURE CLOSETHEWORLD(VAR CH: CHAR);
562   1    7:D     2  VAR
563   1    7:D     2    LTITLE : STRING[29];
564   1    7:D    17    EXITSET: SET OF 'A'..'Z';
565   1    7:0     0  BEGIN
566   1    7:1     0    EXITSET := ['E','E','U','U','R','R'];
567   1    7:1    25    REPEAT
568   1    7:2    25      IF NOT (CH IN EXITSET) THEN BEGIN
569   1    7:4    36        CLEARSCREEN;
570   1    7:4    39        PL:='QUIT: U(PDATE WORK FILE, E(XIT WITHOUT UPDATE, R(ETURN TO EDITOR';
571   1    7:4   111        PROMPT; READ(INPUT,CH); WRITELN(OUTPUT)
572   1    7:3   127      END;
573   1    7:2   127      IF (CH='U') OR (CH='U') THEN
```

```
574   1     7:3   138        WITH USERINFO DO BEGIN
575   1     7:5   138          LTITLE := '*SYSTEM.WRK.TEXT';
576   1     7:5   161          OPENNEW(IOFILE,LTITLE); NEWOUTLOOK;
577   1     7:5   172          (*IF WE GET HERE THEN FILE IS LOCKED ON DISK OK*)
578   1     7:5   172          SYMVID := SYVID; SYMTID := 'SYSTEM.WRK.TEXT'; GOTSYM := TRUE;
579   1     7:5   207          LTITLE := '*SYSTEM.WRK.CODE';
580   1     7:5   230          OPENOLD(IOFILE,LTITLE); CLOSE(IOFILE,PURGE);
581   1     7:5   245          GOTCODE := FALSE; CODETID := ''
582   1     7:4   252      END
583   1     7:1   257    UNTIL CH IN EXITSET;
584   1     7:0   267  END;
585   1     7:0   282
586   1     7:0   282
587   1     8:0     1  PROCEDURE PROMPTS;
588   1     8:0     1  VAR
589   1     8:0     1    HERE: ^INTEGER;
590   1     8:0     0  BEGIN
591   1     8:1     0    MARK(HERE);
592   1     8:1     4    CLEARSCREEN;
593   1     8:1     7    WRITELN(OUTPUT,'YET ANOTHER LINE ORIENTED EDITOR.');
594   1     8:1    56    WRITELN(OUTPUT);
595   1     8:1    62    WRITELN(OUTPUT,
596   1     8:1    62          'ADVANCE  BEGINNING  CHANGE  DELETE  GET  INSERT  JUMP');
597   1     8:1   131    WRITELN(OUTPUT,'KILL  LIST  MACRO <DEFINITION>  NOW <MACRO EXECUTION>');
598   1     8:1   200    WRITELN(OUTPUT,'QUIT <ESC UPDATE>  READ <FILENAME>  SAVE  UNSAVE  VERIFY');
599   1     8:1   272    WRITELN(OUTPUT,'WRITE <FILENAME>  EXCHANGE  ?ELP');
600   1     8:1   320    WRITELN(OUTPUT,'CTRL-X (CAN) TO CANCEL COMMAND INPUT.');
601   1     8:1   373    WRITELN(OUTPUT);
602   1     8:1   379    WRITELN(OUTPUT,'THE MACROS YOU HAVE DEFINED ARE:');
603   1     8:1   427    WRITE(OUTPUT,' - ');
604   1     8:1   440    FOR I := 0 TO MAXMAC DO
605   1     8:2   451      IF MACROS[I].EXEC <> NIL THEN
606   1     8:3   461        WRITE(OUTPUT,I,' - ');
607   1     8:1   489    WRITELN(OUTPUT);
608   1     8:1   495    WRITE(OUTPUT,'YOUR TEXT BUFFER IS ',BUFSIZE,' BYTES, ',ENDPOS);
609   1     8:1   559    WRITELN(OUTPUT,' OF WHICH ARE FILLED, LEAVING ',BUFSIZE-ENDPOS);
610   1     8:1   615    WRITE(OUTPUT,'YOUR ''SAVE'' TEXT IS ',BUFSIZE-BUFEND,' BYTES');
611   1     8:0   671  END;
612   1     8:0   686
613   1     9:0     1  PROCEDURE INCOMMAND;
614   1     9:0     1  LABEL 1,2;
```

```
615   1    9:0      1 VAR ONEESC,WARNED: BOOLEAN;
616   1    9:0      3   CH: CHAR;
617   1    9:0      4   FACTOR,I: INTEGER;
618   1    9:0      6   CHDEL: CHAR;
619   1    9:0      7   CRTESC,UP,EEOL: CHAR;
620   1    9:0     10   SLOW,WASBS: BOOLEAN;
621   1    9:0     12   CONTROLS: SET OF CHAR;
622   1    9:0      0 BEGIN
623   1    9:1      0   FILLCHAR(EXEC^,EXECSIZE,ESC);
624   1    9:1      8   FACTOR := 0;
625   1    9:1     11   WITH SYSCOM^,CRTCTRL,MISCINFO DO
626   1    9:2     28     BEGIN
627   1    9:3     28       SLOW := (BACKSPACE = CHR(0)); (* NO CONTROL *)
628   1    9:3     39       CHDEL := CRTINFO.CHARDEL;
629   1    9:3     48       CRTESC := ESCAPE;
630   1    9:3     55       UP := RLF;
631   1    9:3     64       EEOL := ERASEEOL
632   1    9:2     64     END;
633   1    9:1     73   WASBS := FALSE;
634   1    9:1     76   CH := ' ';
635   1    9:1     79   I := 0;
636   1    9:1     82   WARNED := FALSE;
637   1    9:1     85   ONEESC := FALSE;
638   1    9:1     88   READ(KEYBOARD,CH);
639   1    9:1     96   IF EOLN(KEYBOARD) THEN CH := ACR;
640   1    9:1    109   WHILE (CH <> ESC) OR NOT ONEESC DO
641   1    9:2    117     BEGIN
642   1    9:3    117       IF CH = CHR(SHIFT) THEN
643   1    9:4    122         IF SYSCOM^.MISCINFO.HAS8510A THEN                (*KAS 8/15*)
644   1    9:5    132           IF FACTOR = 128 THEN FACTOR := 0 ELSE FACTOR := 128;
645   1    9:3    149       ONEESC := (CH = ESC);
646   1    9:3    154       IF ONEESC THEN GOTO 1;
647   1    9:3    159       IF CH = CHDEL THEN
648   1    9:4    164             IF (I > 0) THEN
649   1    9:5    169               BEGIN
650   1    9:6    169                 I := PRED(I);
651   1    9:6    174                 IF SLOW THEN
652   1    9:7    177                   IF WASBS THEN WRITE(OUTPUT,EXEC^[I])
653   1    9:7    190                   ELSE WRITE(OUTPUT,'%',EXEC^[I])
654   1    9:6    210                 ELSE
655   1    9:7    212                   IF EXEC^[I] = CHR(TAB) THEN
```

```
656   1     9:8   219                  FOR T := 1 TO 8 DO WRITE(OUTPUT,BACK)
657   1     9:7   239                ELSE WRITE(OUTPUT,BACK,' ',BACK);
658   1     9:5   272              END;
659   1     9:3   272        IF (CH = CHR(CTRLU)) THEN
660   1     9:4   277          BEGIN
661   1     9:5   277            IF SLOW THEN
662   1     9:6   280              WRITELN(OUTPUT,'<ZAP')
663   1     9:5   300            ELSE
664   1     9:6   302              BEGIN
665   1     9:7   302                WRITELN(OUTPUT,CRTESC,UP);
666   1     9:7   324                WRITE(OUTPUT,CRTESC,EEOL);
667   1     9:6   340              END;
668   1     9:5   340            WHILE (I > 0) AND (EXEC^[I] <> ACR) DO I := PRED(I);
669   1     9:5   358            IF I <> 0 THEN I := SUCC(I) ELSE WRITE(OUTPUT,'*')
670   1     9:4   378          END
671   1     9:3   378        ELSE
672   1     9:4   380        IF (CH < ' ') THEN
673   1     9:5   385          BEGIN
674   1     9:6   385            IF ORD(CH) IN [CRET,TAB,DC1] THEN
675   1     9:7   396                BEGIN
676   1     9:8   396          1:                EXEC^[I] := CH;
677   1     9:8   400                          I := SUCC(I);
678   1     9:8   405                          IF ONEESC THEN WRITE(OUTPUT,'$')
679   1     9:8   416                          ELSE IF ORD(CH) = DC1 THEN WRITE(OUTPUT,CHR(7))
680   1     9:9   431                               ELSE WRITE(OUTPUT,CH)
681   1     9:7   441                END;
682   1     9:6   441            IF CH = CHR(CTRLX) THEN BEGIN
683   1     9:8   446                                 I := 0;
684   1     9:8   449                                 WRITELN(OUTPUT);
685   1     9:8   455                                 EXIT(INCOMMAND)
686   1     9:7   459                               END
687   1     9:5   459          END
688   1     9:4   459        ELSE
689   1     9:5   461          IF (CH <> CHDEL) AND WASBS THEN
690   1     9:6   468            BEGIN
691   1     9:7   468              IF SLOW THEN WRITE(OUTPUT,'%');
692   1     9:7   479              WRITE(OUTPUT,CH);
693   1     9:7   487              EXEC^[I] := CH;
694   1     9:7   491              I := SUCC(I)
695   1     9:6   494            END
696   1     9:5   496          ELSE
```

```
697   1    9:6   498            IF (CH <> CHDEL) AND (CH >= ' ') AND (CH <> CHR(CTRLU)) THEN
698   1    9:7   511              BEGIN
699   1    9:8   511                WRITE(OUTPUT,CH);
700   1    9:8   519                EXEC^[I] := CH;
701   1    9:6   523                I := SUCC(I)
702   1    9:7   526              END;
703   1    9:3   528        WASBS := (CH = CHDEL);
704   1    9:3   533        IF I >= (EXECSIZE - 80 (*WARNING*)) THEN
705   1    9:4   542          IF I > (EXECSIZE - 2) THEN REPEAT
706   1    9:6   551            WRITELN(OUTPUT,'COMMAND BUFFER FULL.  TYPE <ESC> <ESC> OR (^X).');
707   1    9:6   614            READ(KEYBOARD,CH);
708   1    9:6   622            IF CH=CHR(CTRLX) THEN BEGIN
709   1    9:8   627              I := 0;  EXIT(INCOMMAND)
710   1    9:7   634            END ELSE
711   1    9:7   636            IF CH = ESC THEN BEGIN
712   1    9:9   641              READ(KEYBOARD,CH);
713   1    9:9   649              IF CH = ESC THEN
714   1    9:0   654                EXIT(INCOMMAND);
715   1    9:8   658            END;
716   1    9:5   658          UNTIL FALSE
717   1    9:4   658          ELSE IF NOT WARNED AND (CH = ACR) THEN
718   1    9:6   671            BEGIN
719   1    9:7   671              WRITELN(OUTPUT,'PLEASE FINISH',CHR(7)(* BELL *));
720   1    9:7   708              WARNED:=TRUE;
721   1    9:6   711            END;
722   1    9:3   711        READ(KEYBOARD,CH);
723   1    9:3   719        IF EOLN(KEYBOARD) THEN CH := ACR;
724   1    9:3   732        IF CH >= ' ' THEN
725   1    9:4   737          CH := CHR(ORD(CH)+FACTOR)
726   1    9:2   740        END;
727   1    9:1   744      WRITELN(OUTPUT,'$');
728   1    9:1   758      I:=I-1;
729   1    9:0   763    END;
730   1    9:0   792
731   1    9:0   792
732   1    2:D     3    FUNCTION COMMAND(*: BOOLEAN *);
733   1    2:D     3      VAR RCOUNT:INTEGER;
734   1    2:D     4        THISCH: CHAR;
735   1    2:D     5        NEG:BOOLEAN;
736   1    2:D     6        NUMBER: SET OF '0'..'9';
737   1    2:D    10
```

```
738   1     2:0     10
739   1    10:0      1 PROCEDURE SYNTAX(ERRCH: CHAR);
740   1    10:0      0 BEGIN
741   1    10:1      0   WRITELN(OUTPUT,ERRCH,' : IS IN ERROR, COMMAND STOPPED.');
742   1    10:1     56   EXIT(COMMAND);
743   1    10:0     60 END;
744   1    10:0     72
745   1    11:0      1 PROCEDURE LINEPLACE(VAR PTR: INTEGER; N: INTEGER);
746   1    11:0      3   VAR I: INTEGER;
747   1    11:0      0 BEGIN
748   1    11:1      0   PTR := CURSOR; (* A NICE PLACE TO START *)
749   1    11:1      3   IF (N <= 0) THEN (* LOOK BACK *) BEGIN
750   1    11:3      8     REPEAT
751   1    11:4      8       PTR := PTR -1;
752   1    11:4     14       I := SCAN(-(PTR+1),=ACR,BUF^[PTR]);
753   1    11:4     29       PTR := PTR +I;
754   1    11:4     35       N := SUCC(N);
755   1    11:3     40     UNTIL (N > 0) OR (PTR < 0);
756   1    11:3     50   PTR := SUCC(PTR);
757   1    11:2     56   END ELSE REPEAT
758   1    11:3     58     I := SCAN(ENDPOS-PTR-1,=ACR,BUF^[PTR]);
759   1    11:3     74     PTR := PTR+I+1;
760   1    11:3     82     N := N -1;
761   1    11:2     87   UNTIL (N=0) OR (PTR = ENDPOS);
762   1    11:0     97 END;
763   1    11:0    114
764   1    12:0      1 PROCEDURE DELETESTUFF;
765   1    12:0      1 VAR
766   1    12:0      1   COUNT: INTEGER;
767   1    12:0      0 BEGIN
768   1    12:1      0   IF (RCOUNT = 0) THEN
769   1    12:2      7     BEGIN
770   1    12:3      7       LINEPLACE(COUNT,0);
771   1    12:3     12       RCOUNT := COUNT - CURSOR;
772   1    12:2     18     END;
773   1    12:1     18   COUNT:=CURSOR+RCOUNT;
774   1    12:1     25   IF RCOUNT<0 THEN
775   1    12:2     32     BEGIN
776   1    12:3     32     IF COUNT<0 THEN COUNT := 0;
777   1    12:3     40     MOVELEFT(BUF^[CURSOR],BUF^[COUNT],ENDPOS-CURSOR+1);
778   1    12:3     51     ENDPOS:=ENDPOS-(CURSOR-COUNT);
```

```
779   1   12:3    58      CURSOR:=COUNT;
780   1   12:2    61    END
781   1   12:1    61    ELSE
782   1   12:2    63      IF (COUNT >= ENDPOS) OR (COUNT < 0) THEN
783   1   12:3    72        BEGIN
784   1   12:4    72          ENDPOS := CURSOR;  BUF^[CURSOR] := CHR(0);
785   1   12:3    79        END
786   1   12:2    79      ELSE
787   1   12:3    81        BEGIN
788   1   12:4    81          MOVELEFT(BUF^[COUNT],BUF^[CURSOR],ENDPOS-COUNT+1);
789   1   12:4    92          ENDPOS:=ENDPOS-(COUNT-CURSOR);
790   1   12:3    99        END;
791   1   12:0    99  END;
792   1   12:0   112
793   1   12:0   112
794   1   13:0     1  PROCEDURE GETTER;
795   1   13:0     1  VAR
796   1   13:0     1    DIR,SIZE: INTEGER;
797   1   13:0     3    FOUND,HARDEND: BOOLEAN;
798   1   13:0     5    FIRST: CHAR;
799   1   13:0     6    PATTERN,QUESTION: STRING[100];
800   1   13:0   108
801   1   14:0     1    PROCEDURE FINDIT;
802   1   14:0     0    BEGIN
803   1   14:1     0      REPEAT
804   1   14:2     0        IF DIR < 0 THEN
805   1   14:3     7          BEGIN
806   1   14:4     7            CURSOR := CURSOR + SCAN(-CURSOR,=FIRST,BUF^[CURSOR]);
807   1   14:4    22            IF CURSOR <= 0 THEN
808   1   14:5    27              BEGIN
809   1   14:6    27                HARDEND := TRUE;
810   1   14:6    31                CURSOR := 0;
811   1   14:6    34                EXIT(FINDIT)
812   1   14:5    38              END
813   1   14:3    38          END
814   1   14:2    38        ELSE
815   1   14:3    40          BEGIN
816   1   14:4    40            CURSOR := CURSOR + SCAN(ENDPOS-CURSOR+1,=FIRST,BUF^[CURSOR]);
817   1   14:4    58            IF CURSOR >= ENDPOS THEN
818   1   14:5    63              BEGIN
819   1   14:6    63                HARDEND := TRUE;
```

```
820   1   14:6    67                CURSOR := ENDPOS;
821   1   14:6    70                EXIT(FINDIT)
822   1   14:5    74              END
823   1   14:3    74            END;
824   1   14:2    74          MOVELEFT(BUF^[CURSOR],QUESTION[1],SIZE);
825   1   14:2    85          FOUND := (QUESTION = PATTERN);
826   1   14:2    96          CURSOR := CURSOR + DIR
827   1   14:1    97        UNTIL FOUND
828   1   14:0   103    END (* FINDIT *);
829   1   14:0   122
830   1   13:0     0  BEGIN
831   1   13:1     0    IF RCOUNT < 0 THEN
832   1   13:2     7      BEGIN
833   1   13:3     7        RCOUNT := -RCOUNT;
834   1   13:3    14        DIR := -1
835   1   13:2    14      END
836   1   13:1    18    ELSE DIR := 1;
837   1   13:1    23    J := J+1;
838   1   13:1    28    SIZE := 0;
839   1   13:1    31    FIRST := EXEC^[J];
840   1   13:1    36    WHILE EXEC^[J +SIZE] <> ESC DO SIZE := SIZE +1;
841   1   13:1    52    IF SIZE >= SIZEOF(PATTERN) THEN
842   1   13:2    57      BEGIN WRITELN(OUTPUT,'FIND TOO LONG'); EXIT(COMMAND) END;
843   1   13:1    90    MOVELEFT(EXEC^[J],PATTERN[1],SIZE);
844   1   13:1    98    PATTERN[0] := CHR(SIZE);
845   1   13:1   103    QUESTION[0] := CHR(SIZE);
846   1   13:1   108    HARDEND := FALSE;
847   1   13:1   111    FOUND := FALSE;
848   1   13:1   114    REPEAT
849   1   13:2   114      FINDIT;
850   1   13:2   116      RCOUNT := RCOUNT -1
851   1   13:1   119    UNTIL (RCOUNT <= 0) OR HARDEND;
852   1   13:1   133    IF HARDEND THEN
853   1   13:2   136      BEGIN
854   1   13:3   136        WRITELN(OUTPUT,PATTERN,' NOT FOUND');
855   1   13:3   171        EXIT(COMMAND)
856   1   13:2   175      END;
857   1   13:1   175    IF DIR < 0 THEN CURSOR := CURSOR +1
858   1   13:1   181    ELSE CURSOR := CURSOR +SIZE -1;
859   1   13:1   194    J := J +SIZE;
860   1   13:1   199    EQUALLENGTH := SIZE
```

```
861   1   13:0   199 END (* GETTER *);
862   1   13:0   218
863   1   15:0     1 PROCEDURE INSERTTEXT;
864   1   15:0     1 VAR
865   1   15:0     1   SIZEOVER: BOOLEAN;  LENGTH,TEMP: INTEGER;
866   1   15:0     0 BEGIN
867   1   15:1     0   SIZEOVER := FALSE;  J := J+1;
868   1   15:1     8   LENGTH := SCAN(I-J,=(ESC),EXEC^[J]);
869   1   15:1    20   TEMP := ENDPOS+LENGTH;
870   1   15:1    25   IF (TEMP > BUFSIZE) THEN
871   1   15:2    30     BEGIN
872   1   15:3    30       WRITELN(OUTPUT,'INSERTION TRUNCATED, NOT ENOUGH SPACE');
873   1   15:3    83       SIZEOVER := TRUE;
874   1   15:3    86       LENGTH := BUFSIZE-ENDPOS;
875   1   15:3    91       TEMP := BUFSIZE;
876   1   15:2    94     END;
877   1   15:1    94   IF (TEMP > BUFEND) THEN
878   1   15:2    99     BEGIN
879   1   15:3    99       WRITELN(OUTPUT,'''SAVE'' AREA DELETED.');
880   1   15:3   135       BUFEND := BUFSIZE;
881   1   15:2   138     END;
882   1   15:1   138   MOVERIGHT(BUF^[CURSOR],BUF^[CURSOR+LENGTH],BUFEND-(CURSOR+LENGTH));
883   1   15:1   151   MOVELEFT(EXEC^[J],BUF^[CURSOR],LENGTH);
884   1   15:1   158   ENDPOS := ENDPOS +LENGTH;
885   1   15:1   163   CURSOR := CURSOR +LENGTH;
886   1   15:1   168   EQUALLENGTH := LENGTH;
887   1   15:1   171   IF SIZEOVER THEN EXIT(COMMAND);
888   1   15:1   178   J := J +LENGTH;
889   1   15:0   183 END (* INSERT NEW TEXT *);
890   1   15:0   196
891   1   16:0     1 PROCEDURE JUMP;
892   1   16:0     0 BEGIN
893   1   16:1     0   IF RCOUNT = 0 THEN LINEPLACE(CURSOR,0)
894   1   16:1    10   ELSE CURSOR := CURSOR + RCOUNT;
895   1   16:1    21   IF (CURSOR<0) AND (RCOUNT<0) THEN CURSOR := 0
896   1   16:1    32   ELSE
897   1   16:2    37    IF (CURSOR<0) OR (CURSOR>ENDPOS) THEN CURSOR := ENDPOS;
898   1   16:0    49 END;
899   1   16:0    62
900   1   16:0    62
901   1   17:0     1 PROCEDURE KILL;
```

```
902   1   17:0     1   VAR POSITION:INTEGER;
903   1   17:0     0 BEGIN
904   1   17:1     0   LINEPLACE(POSITION,RCOUNT);
905   1   17:1     7   IF RCOUNT<=0 THEN
906   1   17:2    14     BEGIN
907   1   17:3    14       MOVELEFT(BUF^[CURSOR],BUF^[POSITION],(ENDPOS-CURSOR+1));
908   1   17:3    25       ENDPOS := ENDPOS - (CURSOR - POSITION);
909   1   17:3    32       CURSOR := CURSOR - (CURSOR - POSITION);
910   1   17:2    39     END
911   1   17:1    39   ELSE
912   1   17:2    41     BEGIN
913   1   17:3    41       MOVELEFT(BUF^[POSITION],BUF^[CURSOR],(ENDPOS-POSITION+1));
914   1   17:3    52       ENDPOS := ENDPOS - (POSITION - CURSOR);
915   1   17:2    59     END;
916   1   17:0    59 END;
917   1   17:0    72
918   1   17:0    72
919   1   18:0     1 PROCEDURE LIST;
920   1   18:0     1   VAR POSITION: INTEGER;
921   1   18:0     0 BEGIN
922   1   18:1     0   LINEPLACE(POSITION,RCOUNT);
923   1   18:1     7   IF RCOUNT<=0 THEN
924   1   18:2    14     UNITWRITE(1(* CONSOLE: *),BUF^[POSITION],CURSOR-POSITION)
925   1   18:1    24   ELSE
926   1   18:2    26     UNITWRITE(1(* CONSOLE: *),BUF^[CURSOR],POSITION-CURSOR)
927   1   18:0    36 END;
928   1   18:0    48
929   1   18:0    48
930   1   19:0     1 PROCEDURE MACRODEFINITION;
931   1   19:0     1 VAR
932   1   19:0     1   STOPCH: CHAR;
933   1   19:0     2   LGTH: INTEGER;
934   1   19:0     0 BEGIN
935   1   19:1     0   IF (RCOUNT<0) OR (RCOUNT>MAXMAC) THEN SYNTAX('#');
936   1   19:1    16   IF MACROS[RCOUNT].EXEC = NIL THEN NEW(MACROS[RCOUNT].EXEC);
937   1   19:1    40   STOPCH := EXEC^[J+1];
938   1   19:1    47   LGTH := SCAN(I-J,=STOPCH,EXEC^[J+2]);
939   1   19:1    61   IF (LGTH = (I-J)) OR (LGTH > SIZEOF(COMARRAY)) OR (LGTH = 0) THEN
940   1   19:2    76     BEGIN
941   1   19:3    76       WRITELN(OUTPUT,'ERROR IN MACRO DEFINITION');
942   1   19:3   117       EXIT(COMMAND);
```

```
943   1   19:2   121     END;
944   1   19:1   121   MOVELEFT(EXEC^[J+2],MACROS[RCOUNT].EXEC^[0],LGTH);
945   1   19:1   157   FILLCHAR(MACROS[RCOUNT].EXEC^[LGTH+1],SIZEOF(COMARRAY)-LGTH,ESC);
946   1   19:1   154   MACROS[RCOUNT].LGTH := LGTH;
947   1   19:1   163   J := J+LGTH+2;
948   1   19:0   170 END (* DEFINE MACRO *);
949   1   19:0   182
950   1   20:0     1 PROCEDURE NOWEXECUTEMACRO;
951   1   20:0     1 VAR
952   1   20:0     1   SAVE: RECORD
953   1   20:0     1             EXEC: ^COMARRAY;
954   1   20:0     1             I,J: INTEGER
955   1   20:0     1           END;
956   1   20:0     4   MACNUM: INTEGER;
957   1   20:0     5   ERROR: BOOLEAN;
958   1   20:0     0 BEGIN
959   1   20:1     0   J := J +1;
960   1   20:1     5   SAVE.EXEC := EXEC;
961   1   20:1     8   SAVE.I := I;
962   1   20:1    11   SAVE.J := J;
963   1   20:1    14   IF EXEC^[J] = ESC THEN MACNUM := 1
964   1   20:1    21   ELSE MACNUM := ORD(EXEC^[J])-ORD('0');
965   1   20:1    33   IF (MACROS[MACNUM].EXEC = NIL) THEN
966   1   20:2    43     BEGIN
967   1   20:3    43       WRITELN(OUTPUT,'ILLEGAL MACRO...TRY AGAIN');
968   1   20:3    84       EXIT(COMMAND)
969   1   20:2    88     END;
970   1   20:1    88   IF (MACNUM<0) OR (MACNUM > MAXMAC) THEN SYNTAX('#');
971   1   20:1   100   EXEC := MACROS[MACNUM].EXEC;
972   1   20:1   108   I := MACROS[MACNUM].LGTH;
973   1   20:1   116   WHILE RCOUNT > 0 DO
974   1   20:2   123     BEGIN
975   1   20:3   123       RCOUNT := RCOUNT -1;
976   1   20:3   131       IF COMMAND THEN
977   1   20:4   137         BEGIN
978   1   20:5   137           COMMAND := TRUE;
979   1   20:5   141           EXIT(COMMAND)
980   1   20:4   145         END;
981   1   20:3   145       ERROR := (J<I);
982   1   20:3   150       IF ERROR THEN
983   1   20:4   153         BEGIN
```

```
 984   1    20:5   153            RCOUNT := 0;
 985   1    20:5   157            WRITELN(OUTPUT,'MACRO HALTED');
 986   1    20:4   185          END;
 987   1    20:2   188        END;
 988   1    20:1   197      EXEC := SAVE.EXEC;
 989   1    20:1   190      I := SAVE.I;
 990   1    20:1   193      J := SAVE.J;
 991   1    20:1   196      IF ERROR THEN EXIT(COMMAND);
 992   1    20:0   203  END (* NOW EXECUTE MACRO *);
 993   1    20:0   218
 994   1    21:0     1  PROCEDURE OPTIONMOD;
 995   1    21:0     0  BEGIN
 996   1    21:1     0    WITH OPTION DO
 997   1    21:2     0      BEGIN
 998   1    21:3     0        ONOFF := NOT ONOFF;
 999   1    21:3    11        IF ONOFF THEN
1000   1    21:4    18          WITH SYSCOM^.CRTINFO DO
1001   1    21:5    25            IF RCOUNT > 1 THEN
1002   1    21:6    32              LISTSIZE := RCOUNT
1003   1    21:5    36            ELSE
1004   1    21:6    42              LISTSIZE := HEIGHT DIV 2 -1
1005   1    21:2    50      END
1006   1    21:0    53  END;
1007   1    21:0    56
1008   1    22:0     1  PROCEDURE READFILE;
1009   1    22:0     1  VAR
1010   1    22:0     1    LGTH: INTEGER;
1011   1    22:0     2    TITLE: STRING[40];
1012   1    22:0     0  BEGIN
1013   1    22:1     0    J := J +1;
1014   1    22:1     5    LGTH := SCAN(30,=ESC,EXEC^[J]);
1015   1    22:1    15    IF (LGTH <= 30) AND (LGTH > 0) THEN
1016   1    22:2    24      BEGIN
1017   1    22:3    24        TITLE[0] := CHR(LGTH);
1018   1    22:3    29        MOVELEFT(EXEC^[J],TITLE[1],LGTH);
1019   1    22:3    37        OPENOLD(IOFILE,TITLE);
1020   1    22:3    46        IF IORESULT = 0 THEN
1021   1    22:4    52          BEGIN IF NEWFIN THEN EXIT(COMMAND) END
1022   1    22:3    62        ELSE
1023   1    22:4    64          BEGIN
1024   1    22:5    64            OPENOLD(IOFILE,CONCAT(TITLE,'.TEXT'));
```

```
1025   1   22:5    95            IF IORESULT = 0 THEN
1026   1   22:6   104              BEGIN IF NEWFIN THEN EXIT(COMMAND) END
1027   1   22:5   114            ELSE
1028   1   22:6   115              BEGIN
1029   1   22:7   119                WRITELN(OUTPUT,'FILE: ',TITLE,' IS IN ERROR.  NOT READ');
1030   1   22:7   130                EXIT(COMMAND);
1031   1   22:6   134              END;
1032   1   22:4   164          END
1033   1   22:2   164      END
1034   1   22:1   164    ELSE
1035   1   22:2   186      BEGIN
1036   1   22:3   186        WRITELN(OUTPUT,'FILE NAME ERROR.');
1037   1   22:3   218        EXIT(COMMAND);
1038   1   22:2   222      END;
1039   1   22:1   222    J := J +LGTH;
1040   1   22:0   227  END;
1041   1   22:0   242
1042   1   23:0     1  PROCEDURE SAVE;
1043   1   23:0     1  VAR
1044   1   23:0     1    POS,DELTA: INTEGER;
1045   1   23:0     0  BEGIN
1046   1   23:1     0    LINEPLACE(POS,RCOUNT);
1047   1   23:1     7    IF RCOUNT <= 0 THEN
1048   1   23:2    14      DELTA := CURSOR -POS
1049   1   23:1    15    ELSE
1050   1   23:2    21      DELTA := POS -CURSOR;
1051   1   23:1    26    BUFEND := BUFSIZE -DELTA;
1052   1   23:1    31    IF BUFEND <= ENDPOS THEN
1053   1   23:2    36      BEGIN
1054   1   23:3    36        BUFEND := BUFSIZE;
1055   1   23:3    39        WRITELN(OUTPUT,'NOT ENOUGH ROOM TO SAVE IN');
1056   1   23:3    81        EXIT(COMMAND);
1057   1   23:2    85      END;
1058   1   23:1    85    IF RCOUNT <= 0 THEN
1059   1   23:2    92      MOVELEFT(BUF^[POS],BUF^[BUFEND],DELTA)
1060   1   23:1    99    ELSE
1061   1   23:2   101      MOVELEFT(BUF^[CURSOR],BUF^[BUFEND],DELTA)
1062   1   23:0   108  END (* SAVE *);
1063   1   23:0   120
1064   1   24:0     1  PROCEDURE UNSAVE;
1065   1   24:0     1  VAR
```

```
1066   1    24:0      1   STASHSIZE,STASHEDAT,DELTA: INTEGER;
1067   1    24:0      0 BEGIN
1068   1    24:1      0   IF RCOUNT = 0 THEN
1069   1    24:2      7     BUFEND := BUFSIZE
1070   1    24:1      7   ELSE
1071   1    24:2     12     BEGIN
1072   1    24:3     12       STASHSIZE := ENDPOS -CURSOR;
1073   1    24:3     17       DELTA := BUFSIZE -BUFEND;
1074   1    24:3     22       STASHEDAT := CURSOR +DELTA;
1075   1    24:3     27       IF ((STASHEDAT +STASHSIZE) < BUFEND) THEN
1076   1    24:4     34         BEGIN
1077   1    24:5     34           MOVERIGHT(BUF^[CURSOR],BUF^[STASHEDAT],STASHSIZE);
1078   1    24:5     41           MOVELEFT(BUF^[BUFEND],BUF^[CURSOR],DELTA);
1079   1    24:5     48           ENDPOS := ENDPOS +DELTA;
1080   1    24:5     53           BUF^[ENDPOS] := CHR(0)
1081   1    24:4     56         END
1082   1    24:3     57       ELSE
1083   1    24:4     59         BEGIN WRITELN(OUTPUT,'NOT ENOUGH SPACE'); EXIT(COMMAND) END
1084   1    24:2     95     END (* ^=0 *)
1085   1    24:0     95 END (* UNSAVE *);
1086   1    24:0    108
1087   1    25:0      1 PROCEDURE VIEW;
1088   1    25:0      0 BEGIN
1089   1    25:1      0   RCOUNT := 0;  LIST;
1090   1    25:1      6   RCOUNT := 1;  LIST
1091   1    25:0     10 END;
1092   1    25:0     24
1093   1    26:0      1 PROCEDURE WRITEFILE;
1094   1    26:0      1 VAR
1095   1    26:0      1   LGTH: INTEGER;
1096   1    26:0      2   TITLE: STRING[40];
1097   1    26:0      0 BEGIN
1098   1    26:1      0   J := J +1;
1099   1    26:1      5   LGTH := SCAN(30,=ESC,EXEC^[J]);
1100   1    26:1     15   IF (LGTH > 0) AND (LGTH <= 30) THEN BEGIN
1101   1    26:3     24     TITLE[0] := CHR(LGTH);
1102   1    26:3     29     MOVELEFT(EXEC^[J],TITLE[1],LGTH);
1103   1    26:3     37     IF (TITLE[LGTH] <> '.') AND (TITLE[LGTH] <> ']') AND
1104   1    26:3     50         (POS('.TEXT',TITLE) = 0) THEN
1105   1    26:4     70       TITLE := CONCAT(TITLE,'.TEXT');
1106   1    26:3    101     IF (TITLE[LGTH] = '.') THEN DELETE(TITLE,LGTH,1);
```

```
1107   1   26:3   116      OPENNEW(IOFILE,TITLE);
1108   1   26:3   126      IF IORESULT = 0 THEN
1109   1   26:4   131        NEWOUTLOOK
1110   1   26:3   131      ELSE BEGIN
1111   1   26:5   135        WRITELN(OUTPUT,CONCAT('FILE: ',TITLE,' IS IN ERROR. WRITE NOT DONE.'));
1112   1   26:5   214        EXIT(COMMAND);
1113   1   26:4   218      END;
1114   1   26:2   218    END ELSE BEGIN
1115   1   26:3   220      WRITELN(OUTPUT,'ILLEGAL TITLE');
1116   1   26:3   249      EXIT(COMMAND);
1117   1   26:2   253    END;
1118   1   26:1   253    J := J +LGTH;
1119   1   26:0   258  END;
1120   1   26:0   272
1121   1    2:0     0  BEGIN (*COMMAND*)
1122   1    2:1     0    COMMAND := FALSE;
1123   1    2:1     3    NUMBER := ['0'..'9'];
1124   1    2:1    21    J := 0;
1125   1    2:1    24    WHILE (J<I) DO
1126   1    2:2    29      BEGIN
1127   1    2:3    29        WHILE (EXEC^[J] IN [' ',ACR,CHR(TAB),ESC]) AND (J<I) DO
1128   1    2:4    57          J := SUCC(J);
1129   1    2:3    64        THISCH := EXEC^[J];
1130   1    2:3    69        NEG := (THISCH = '-');
1131   1    2:3    74        IF THISCH IN ['+','-'] THEN
1132   1    2:4    88          BEGIN
1133   1    2:5    88            J := J +1;
1134   1    2:5    93            THISCH := EXEC^[J]
1135   1    2:4    95          END;
1136   1    2:3    98        IF (THISCH IN NUMBER) THEN
1137   1    2:4   107          BEGIN
1138   1    2:5   107            RCOUNT := 0;
1139   1    2:5   110            REPEAT
1140   1    2:6   110              RCOUNT := (RCOUNT*10) + ORD(EXEC^[J])-ORD('0');
1141   1    2:6   121              J := SUCC(J)
1142   1    2:5   124            UNTIL ((NOT (EXEC^[J] IN NUMBER)) OR (RCOUNT > 3200));
1143   1    2:5   144            THISCH := EXEC^[J];
1144   1    2:4   149          END(* IN NUMBER *)
1145   1    2:3   149        ELSE RCOUNT := 1;
1146   1    2:3   154        IF (THISCH IN ['=','/']) THEN
1147   1    2:4   170          IF (RCOUNT <> 1) THEN SYNTAX(THISCH)
```

```
1148   1     2:4    176          ELSE
1149   1     2:5    180            BEGIN
1150   1     2:6    180              IF (THISCH = '=') THEN RCOUNT := -EQUALLENGTH
1151   1     2:6    185              ELSE (* = '/' *) RCOUNT := 32700;
1152   1     2:6    196              J := J +1;
1153   1     2:6    201              THISCH := EXEC^[J]
1154   1     2:5    203            END;
1155   1     2:3    206        IF NEG THEN RCOUNT := -RCOUNT;
1156   1     2:3    213        IF (J >= I) THEN EXIT(COMMAND);
1157   1     2:3    222        IF (THISCH IN ['?','A'..'Z','A'..'Z']) THEN
1158   1     2:4    246          CASE THISCH OF
1159   1     2:4    249            '?'     : PROMPTS;
1160   1     2:4    253            'A','A':LINEPLACE(CURSOR,RCOUNT);
1161   1     2:4    260            'B','B':CURSOR:=0; (*DA END*)
1162   1     2:4    265            'C','C':BEGIN DELETESTUFF; INSERTTEXT END;
1163   1     2:4    271            'D','D':DELETESTUFF;
1164   1     2:4    275            'E','E':CLEARSCREEN;
1165   1     2:4    280            'G','F','F','G':GETTER;
1166   1     2:4    284            'H','H':WRITELN(OUTPUT,'UNIMPLEMENTED');
1167   1     2:4    315            'I','I': INSERTTEXT;
1168   1     2:4    319            'J','J':JUMP;
1169   1     2:4    323            'K','K':KILL;
1170   1     2:4    327            'L','L':LIST;
1171   1     2:4    331            'M','M': MACRODEFINITION;
1172   1     2:4    335            'N','N': NOWEXECUTEMACRO;
1173   1     2:4    339            'O','O': OPTIONMOD;
1174   1     2:4    343            'P','T','Y','Z',
1175   1     2:4    343            'P','T','Y','Z': SYNTAX(THISCH);
1176   1     2:4    348            'Q','Q': BEGIN
1177   1     2:6    348                     THISCH := EXEC^[J+1];
1178   1     2:6    355                     CLOSETHEWORLD(THISCH);
1179   1     2:6    359                     COMMAND := (THISCH IN ['E','E','U','U']);
1180   1     2:6    382                     EXIT(COMMAND)
1181   1     2:5    386                   END;
1182   1     2:4    388            'R','R':READFILE;
1183   1     2:4    392            'S','S':SAVE;
1184   1     2:4    396            'U','U':UNSAVE;
1185   1     2:4    400            'V','V':VIEW;
1186   1     2:4    404            'W','W':WRITEFILE;
1187   1     2:4    408            'X','X':BEGIN KILL; INSERTTEXT END
1188   1     2:4    412          END
```

```
1139   1     2:3    542        ELSE SYNTAX(THISCH);
1190   1     2:3    547        J:=J+1;
1191   1     2:2    552      END (* WHILE J <= I *);
1192   1     2:1    554    IF OPTION.ONOFF THEN
1193   1     2:2    561      BEGIN
1194   1     2:3    561        CLEARSCREEN;
1195   1     2:3    564        RCOUNT := -OPTION.LISTSIZE;
1196   1     2:3    572        LIST;
1197   1     2:3    574        WRITE(OUTPUT,CHR(10 (* LF *)));
1198   1     2:3    582        RCOUNT := OPTION.LISTSIZE;
1199   1     2:3    589        LIST
1200   1     2:2    589      END;
1201   1     2:3    591 END (* COMMAND *);
1202   1     2:3    618
1203   1     1:0      0 BEGIN (*YALOE*)
1204   1     1:1      3   INITIALIZE;
1205   1     1:1     14   REPEAT
1206   1     1:2     14     WRITE(KEYBOARD,'*'); (*CLEARS ^F AND ^S FLAGS!*)
1207   1     1:2     22 (*    THIS LINE IS FOR THE HAVAHEART COMMAND
1208   1     1:2     22  *  MOVELEFT(EXEC^,BUF^[ENDPOS+1],MIN(I,BUFEND-ENDPOS));
1209   1     1:2     22  *         WHICH SOME DAY MAY BE IMPLEMENTED *)
1210   1     1:2     22     INCOMMAND
1211   1     1:1     22   UNTIL COMMAND;
1212   1     1:0     30 END;
1213   1     1:0     50
1214   0     1:0      0 BEGIN (* JUST A DUMMY *)
1215   0     1:0      0 END.
```

```
  1    1     1:0     1 (*$L PRINTER:*)
  2    1     1:0     1 (* SWAPPING PASCAL COMPILER INCLUDE FILES *)
  3    1     1:0     1 (*$C COPYRIGHT (C) 1979 REGENTS UCSD II.0.A.1*)
  4    1     1:0     1 (*$T+*) (*$S+*)
  4    1     1:0     1 (*$I #5:COMPGLBLS.TEXT*)
  5    1     1:0     1 (*$U-*)
  6    0     1:0     1 PROGRAM PASCALSYSTEM;  (* VERSION II.0  1-31-79 *)
  7    0     1:0     1
  8    0     1:0     1
  9    0     1:0     1 (************************************************)
 10    0     1:0     1 (*                                              *)
 11    0     1:0     1 (*         UCSD  PASCAL  COMPILER               *)
 12    0     1:0     1 (*                                              *)
 13    0     1:0     1 (*     BASED ON ZURICH P2 PORTABLE              *)
 14    0     1:0     1 (*     COMPILER, EXTENSIVLY                     *)
 15    0     1:0     1 (*     MODIFIED BY ROGER T. SUMNER              *)
 16    0     1:0     1 (*     SHAWN FANNING AND ALBERT A. HOFFMAN      *)
 17    0     1:0     1 (*     1976..1979                               *)
 18    0     1:0     1 (*                                              *)
 19    0     1:0     1 (*     RELEASE LEVEL: I.3 AUGUST, 1977          *)
 20    0     1:0     1 (*                    I.4 JANUARY, 1978         *)
 21    0     1:0     1 (*                    I.5 SEPTEMBER, 1978       *)
 22    0     1:0     1 (*                   II.0 JANUARY, 1979         *)
 23    0     1:0     1 (*                                              *)
 24    0     1:0     1 (*     INSTITUTE FOR INFORMATION SYSTEMS        *)
 25    0     1:0     1 (*     UC SAN DIEGO, LA JOLLA, CA 92093         *)
 26    0     1:0     1 (*                                              *)
 27    0     1:0     1 (*     KENNETH L. BOWLES, DIRECTOR              *)
 28    0     1:0     1 (*                                              *)
 29    0     1:0     1 (*     COPYRIGHT (C) 1979, REGENTS OF THE       *)
 30    0     1:0     1 (*     UNIVERSITY OF CALIFORNIA, SAN DIEGO      *)
 31    0     1:0     1 (*                                              *)
 32    0     1:0     1 (************************************************)
 33    0     1:0     1 TYPE PHYLE = FILE;
 34    0     1:0     1 INFOREC = RECORD
 35    0     1:0     1             WORKSYM,WORKCODE: ^PHYLE;
 36    0     1:0     1             ERRSYM,ERRBLK,ERRNUM: INTEGER;
 37    0     1:0     1             SLOWTERM,STUPID: BOOLEAN;
 38    0     1:0     1             ALTMODE: CHAR
 39    0     1:0     1           END;
 40    0     1:0     1
```

```
41    0     1:0     1
42    0     1:0     1
43    1     1:0     1 SEGMENT PROCEDURE USERPROGRAM;
44    1     1:0     1
45    2     1:0     1   SEGMENT PROCEDURE FILEHANDLER;
46    2     1:0     0   BEGIN END;
47    2     1:0    12
48    3     1:0     1   SEGMENT PROCEDURE DEBUGGER;
49    3     1:0     0   BEGIN END;
50    3     1:0    12
51    4     1:0     1   SEGMENT PROCEDURE PRINTERROR;
52    4     1:0     0   BEGIN END;
53    4     1:0    12
54    5     1:0     1   SEGMENT PROCEDURE INITIALIZE;
55    5     1:0     0   BEGIN END;
56    5     1:0    12
57    6     1:0     1   SEGMENT PROCEDURE GETCMD;
58    6     1:0     0   BEGIN END;
59    6     1:0    12
60    7     1:0     1   SEGMENT PROCEDURE NOTUSED1;
61    7     1:0     0   BEGIN END;
62    7     1:0    12
63    8     1:0     1   SEGMENT PROCEDURE NOTUSED2;
64    8     1:0     0   BEGIN END;
65    8     1:0    12
66    9     1:0     1   SEGMENT PROCEDURE NOTUSED3;
67    9     1:0     0   BEGIN END;
68    9     1:0    12
69    1     1:0     0 BEGIN END; (* USERPROGRAM *)
70    1     1:0    12
71   10     1:0     1 SEGMENT PROCEDURE PASCALCOMPILER(VAR USERINFO: INFOREC);
72   10     1:0     2
73   10     1:0     2 CONST DISPLIMIT = 12; MAXLEVEL = 8; MAXADDR = 28000;
74   10     1:0     2       INTSIZE = 1; REALSIZE = 2; BITSPERWD = 16;
75   10     1:0     2       CHARSIZE = 1; BOOLSIZE = 1; PTRSIZE = 1;
76   10     1:0     2       FILESIZE = 300; NILFILESIZE = 40; BITSPERCHR = 8; CHRSPERWD = 2;
77   10     1:0     2       STRINGSIZE = 0; STRGLGTH = 255; MAXINT = 32767; MAXDEC = 36;
78   10     1:0     2       DEFSTRGLGTH = 80; LCAFTERMARKSTACK = 1; REFSPERBLK = 128;
79   10     1:0     2       EOL = 13; MAXCURSOR = 1023; MAXCODE = 1299;
80   10     1:0     2       MAXJTAB = 24; MAXSEG = 15; MAXPROCNUM = 149;
81   10     1:0     2
```

```
 82  10     1:D      2 TYPE
 83  10     1:D      2                    (*BASIC SYMBOLS, MUST MATCH ORDER IN IDSEARCH*)
 84  10     1:D      2
 85  10     1:D      2      SYMBOL =  (IDENT,COMMA,COLON,SEMICOLON,LPARENT,RPARENT,DOSY,TOSY,
 86  10     1:D      2                DOWNTOSY,ENDSY,UNTILSY,OFSY,THENSY,ELSESY,BECOMES,LBRACK,
 87  10     1:D      2                RBRACK,ARROW,PERIOD,BEGINSY,IFSY,CASESY,REPEATSY,WHILESY,
 88  10     1:D      2                FORSY,WITHSY,GOTOSY,LABELSY,CONSTSY,TYPESY,VARSY,PROCSY,
 89  10     1:D      2                FUNCSY,PROGSY,FORWARDSY,INTCONST,REALCONST,STRINGCONST,
 90  10     1:D      2                NOTSY,MULOP,ADDOP,RELOP,SETSY,PACKEDSY,ARRAYSY,RECORDSY,
 91  10     1:D      2                FILESY,OTHERSY,LONGCONST,USESSY,UNITSY,INTERSY,IMPLESY,
 92  10     1:D      2                EXTERNLSY,SEPARATSY);
 93  10     1:D      2
 94  10     1:D      2
 95  10     1:D      2      OPERATOR = (MUL,RDIV,ANDOP,IDIV,IMOD,PLUS,MINUS,OROP,LTOP,LEOP,
 96  10     1:D      2                 GEOP,GTOP,NEOP,EQOP,INOP,NOOP);
 97  10     1:D      2
 98  10     1:D      2      SETOFSYS = SET OF SYMBOL;
 99  10     1:D      2
100  10     1:D      2      NONRESIDENT = (SEEK,FREADREAL,FWRITEREAL,FREADDEC,FWRITEDEC,DECOPS);
101  10     1:D      2      NONRESPFLIST = ARRAY[NONRESIDENT] OF INTEGER;
102  10     1:D      2
103  10     1:D      2                                                    (*CONSTANTS*)
104  10     1:D      2      CSTCLASS = (REEL,PSET,STRG,TRIX,LONG);
105  10     1:D      2      CSP = ^ CONSTREC;
106  10     1:D      2      CONSTREC = RECORD CASE CCLASS: CSTCLASS OF
107  10     1:D      2                          LONG: (LLENG,LLAST: INTEGER;
108  10     1:D      2                                 LONGVAL: ARRAY[1..9] OF INTEGER);
109  10     1:D      2                          TRIX: (CSTVAL: ARRAY [1..8] OF INTEGER);
110  10     1:D      2                                 (*MUST COMPLETELY OVERLAP FOLLOWING FIELDS*)
111  10     1:D      2                          REEL: (RVAL: REAL);
112  10     1:D      2                          PSET: (PVAL: SET OF 0..127);
113  10     1:D      2                          STRG: (SLGTH: 0..STRGLGTH;
114  10     1:D      2                                 SVAL: PACKED ARRAY [1..STRGLGTH] OF CHAR)
115  10     1:D      2                        END;
116  10     1:D      2
117  10     1:D      2      VALU = RECORD CASE BOOLEAN OF
118  10     1:D      2                      TRUE:  (IVAL: INTEGER);
119  10     1:D      2                      FALSE: (VALP: CSP)
120  10     1:D      2                    END;
121  10     1:D      2
122  10     1:D      2                                                    (*DATA STRUCTURES*)
```

```
123  10    1:0     2      BITRANGE = 0..BITSPERWD; OPRANGE = 0..80;
124  10    1:0     2      CURSRANGE = 0..MAXCURSOR; PROCRANGE = 0..MAXPROCNUM;
125  10    1:0     2      LEVRANGE = 0..MAXLEVEL; ADDRRANGE = 0..MAXADDR;
126  10    1:0     2      JTABRANGE = 0..MAXJTAB; SEGRANGE = 0..MAXSEG;
127  10    1:0     2      DISPRANGE = 0..DISPLIMIT;
128  10    1:0     2
129  10    1:0     2      STRUCTFORM = (SCALAR,SUBRANGE,POINTER,LONGINT,POWER,ARRAYS,
130  10    1:0     2                    RECORDS,FILES,TAGFLD,VARIANT);
131  10    1:0     2
132  10    1:0     2      DECLKIND = (STANDARD,DECLARED,SPECIAL);
133  10    1:0     2
134  10    1:0     2      STP = ^ STRUCTURE; CTP = ^ IDENTIFIER;
135  10    1:0     2
136  10    1:0     2      STRUCTURE = RECORD
137  10    1:0     2                    SIZE: ADDRRANGE;
138  10    1:0     2                    CASE FORM: STRUCTFORM OF
139  10    1:0     2                      SCALAR:    (CASE SCALKIND: DECLKIND OF
140  10    1:0     2                                    DECLARED: (FCONST: CTP));
141  10    1:0     2                      SUBRANGE:  (RANGETYPE: STP; MIN,MAX: VALU);
142  10    1:0     2                      POINTER:   (ELTYPE: STP);
143  10    1:0     2                      POWER:     (ELSET: STP);
144  10    1:0     2                      ARRAYS:    (AELTYPE,INXTYPE: STP;
145  10    1:0     2                                  CASE AISPACKD:BOOLEAN OF
146  10    1:0     2                                    TRUE: (ELSPERWD,ELWIDTH: BITRANGE;
147  10    1:0     2                                           CASE AISSTRNG: BOOLEAN OF
148  10    1:0     2                                            TRUE:(MAXLENG: 1..STRGLGTH)));
149  10    1:0     2                      RECORDS:   (FSTFLD: CTP; RECVAR: STP);
150  10    1:0     2                      FILES:     (FILTYPE: STP);
151  10    1:0     2                      TAGFLD:    (TAGFIELDP: CTP; FSTVAR: STP);
152  10    1:0     2                      VARIANT:   (NXTVAR,SUBVAR: STP; VARVAL: VALU)
153  10    1:0     2                    END;
154  10    1:0     2
155  10    1:0     2                                                          (*NAMES*)
156  10    1:0     2      IDCLASS = (TYPES,KONST,FORMALVARS,ACTUALVARS,FIELD,
157  10    1:0     2                 PROC,FUNC,MODULE);
158  10    1:0     2      SETOFIDS = SET OF IDCLASS;
159  10    1:0     2      IDKIND = (ACTUAL,FORMAL);
160  10    1:0     2      ALPHA = PACKED ARRAY [1..8] OF CHAR;
161  10    1:0     2
162  10    1:0     2      IDENTIFIER = RECORD
163  10    1:0     2                    NAME: ALPHA; LLINK, RLINK: CTP;
```

```
164  10    1:D     2                IDTYPE: STP; NEXT: CTP;
165  10    1:D     2                CASE KLASS: IDCLASS OF
166  10    1:D     2                  KONST: (VALUES: VALU);
167  10    1:D     2              FORMALVARS,
168  10    1:D     2              ACTUALVARS: (VLEV: LEVRANGE;
169  10    1:D     2                           VADDR: ADDRRANGE;
170  10    1:D     2                           CASE BOOLEAN OF
171  10    1:D     2                             TRUE: (PUBLIC: BOOLEAN));
172  10    1:D     2                FIELD: (FLDADDR: ADDRRANGE;
173  10    1:D     2                        CASE FISPACKD: BOOLEAN OF
174  10    1:D     2                          TRUE: (FLDRBIT,FLDWIDTH: BITRANGE));
175  10    1:D     2                PROC,
176  10    1:D     2                FUNC:  (CASE PFDECKIND: DECLKIND OF
177  10    1:D     2                         SPECIAL:  (KEY: INTEGER);
178  10    1:D     2                         STANDARD: (CSPNUM: INTEGER);
179  10    1:D     2                         DECLARED: (PFLEV: LEVRANGE;
180  10    1:D     2                                    PFNAME: PROCRANGE;
181  10    1:D     2                                    PFSEG: SEGRANGE;
182  10    1:D     2                                    CASE PFKIND: IDKIND OF
183  10    1:D     2                                     ACTUAL: (LOCALLC: ADDRRANGE;
184  10    1:D     2                                              FORWDECL: BOOLEAN;
185  10    1:D     2                                              EXTURNAL: BOOLEAN;
186  10    1:D     2                                              INSCOPE: BOOLEAN;
187  10    1:D     2                                              CASE BOOLEAN OF
188  10    1:D     2                                               TRUE: (IMPORTED:BOOLEAN))));
189  10    1:D     2              MODULE: (SEGID: INTEGER)
190  10    1:D     2              END;
191  10    1:D     2
192  10    1:D     2
193  10    1:D     2    WHERE = (BLCK,CREC,VREC,REC);
194  10    1:D     2
195  10    1:D     2
196  10    1:D     2                                          (*EXPRESSIONS*)
197  10    1:D     2    ATTRKIND = (CST,VARBL,EXPR);
198  10    1:D     2    VACCESS = (DRCT,INDRCT,PACKD,MULTI,BYTE);
199  10    1:D     2
200  10    1:D     2    ATTR = RECORD TYPTR: STP;
201  10    1:D     2             CASE KIND: ATTRKIND OF
202  10    1:D     2               CST:   (CVAL: VALU);
203  10    1:D     2               VARBL: (CASE ACCESS: VACCESS OF
204  10    1:D     2                         DRCT:   (VLEVEL: LEVRANGE; DPLMT: ADDRRANGE);
                                           INDRCT: (IDPLMT: ADDRRANGE))
```

```
205  10    1:D    2              END;
206  10    1:D    2
207  10    1:D    2      TESTP = ^ TESTPOINTER;
208  10    1:D    2      TESTPOINTER = RECORD
209  10    1:D    2                      ELT1,ELT2 : STP;
210  10    1:D    2                      LASTTESTP : TESTP
211  10    1:D    2                    END;
212  10    1:D    2
213  10    1:D    2                                                    (*LABELS*)
214  10    1:D    2      LBP = ^ CODELABEL;
215  10    1:D    2      CODELABEL = RECORD
216  10    1:D    2                    CASE DEFINED: BOOLEAN OF
217  10    1:D    2                      FALSE: (REFLIST: ADDRRANGE);
218  10    1:D    2                      TRUE:  (OCCURIC: ADDRRANGE; JTABINX: JTABRANGE)
219  10    1:D    2                  END;
220  10    1:D    2
221  10    1:D    2      LABELP = ^ USERLABEL;
222  10    1:D    2      USERLABEL = RECORD
223  10    1:D    2                    LABVAL: INTEGER;
224  10    1:D    2                    NEXTLAB: LABELP;
225  10    1:D    2                    CODELBP: LBP
226  10    1:D    2                  END;
227  10    1:D    2
228  10    1:D    2      REFARRAY = ARRAY[1..REFSPERBLK] OF
229  10    1:D    2                   RECORD
230  10    1:D    2                     KEY,OFFSET: INTEGER
231  10    1:D    2                   END;
232  10    1:D    2
233  10    1:D    2      CODEARRAY = PACKED ARRAY [0..MAXCODE] OF CHAR;
234  10    1:D    2      SYMBUFARRAY = PACKED ARRAY [CURSRANGE] OF CHAR;
235  10    1:D    2
236  10    1:D    2      UNITFILE = (WORKCODE,SYSLIBRARY);
237  10    1:D    2
238  10    1:D    2      LEXSTKREC = RECORD
239  10    1:D    2                    DOLDTOP: DISPRANGE;
240  10    1:D    2                    DOLDLEV: 0..MAXLEVEL;
241  10    1:D    2                    POLDPROC,SOLDPROC: PROCRANGE;
242  10    1:D    2                    DOLDSEG: SEGRANGE;
243  10    1:D    2                    DLLC: ADDRRANGE;
244  10    1:D    2                    BFSY: SYMBOL;
245  10    1:D    2                    DFPROCP: CTP;
```

```
246  10    1:D     2                    DMARKP: ^INTEGER;
247  10    1:D     2                    ISSEGMENT: BOOLEAN;
248  10    1:D     2                    PREVLEXSTACKP: ^LEXSTKREC
249  10    1:D     2                  END;
250  10    1:D     2
251  10    1:D     2
252  10    1:D     2 (*-------------------------------------------------------------*)
253  10    1:D     2
254  10    1:D     2 VAR
255  10    1:D     2
256  10    1:D     2     CODEP: ^ CODEARRAY;              (*CODE BUFFER UNTIL WRITEOUT*)
257  10    1:D     3     SYMBUFP: ^ SYMBUFARRAY;          (*SYMBOLIC BUFFER...ASCII OR CODED*)
258  10    1:D     4
259  10    1:D     4     GATTR: ATTR;                     (*DESCRIBES CURRENT EXPRESSION*)
260  10    1:D     9
261  10    1:D     9     TOP: DISPRANGE;                  (*TOP OF DISPLAY*)
262  10    1:D    10     LC,IC: ADDRRANGE;                (*LOCATION AND INSTRUCT COUNTERS*)
263  10    1:D    12     TEST: BOOLEAN;
264  10    1:D    13     INTPTR: STP;                     (*POINTER TO STANDARD INTEGER TYPE*)
265  10    1:D    14     SEG: SEGRANGE;                   (*CURRENT SEGMENT NO.*)
266  10    1:D    15                                      (*SCANNER GLOBALS...NEXT FOUR VARS*)
267  10    1:D    15                                      (*MUST BE IN THIS ORDER FOR IDSEARCH*)
268  10    1:D    15     SYMCURSOR: CURSRANGE;            (*CURRENT SCANNING INDEX IN SYMBUFP^*)
269  10    1:D    16     SY: SYMBOL;                      (*SYMBOL FOUND BY INSYMBOL*)
270  10    1:D    17     OP: OPERATOR;                    (*CLASSIFICATION OF LAST SYMBOL*)
271  10    1:D    18     ID: ALPHA;                       (*LAST IDENTIFIER FOUND*)
272  10    1:D    22
273  10    1:D    22     LGTH: INTEGER;                   (*LENGTH OF LAST STRING CONSTANT IN CHARS
274  10    1:D    23                                        OR LEN OF LAST LONG INTEGER CONSTANT
275  10    1:D    23                                        IN DIGITS*)
276  10    1:D    23     VAL: VALU;                       (*VALUE OF LAST CONSTANT*)
277  10    1:D    24     DISX: DISPRANGE;                 (*LEVEL OF LAST ID SEARCHED*)
278  10    1:D    25
279  10    1:D    25     LCMAX: ADDRRANGE;                (*TEMPORARIES LOCATION COUNTER*)
280  10    1:D    26
281  10    1:D    26                                      (*SWITCHES:*)
282  10    1:D    26
283  10    1:D    26     PRTERR,GOTOOK,RANGECHECK,DEBUGGING,
284  10    1:D    26     NOISY,CODEINSEG,IOCHECK,BPTONLINE,
285  10    1:D    26     CLINKERINFO,DLINKERINFO,LIST,TINY,LSEPPROC,
286  10    1:D    26     DP,INCLUDING,USING,NOSWAP,SEPPROC,
```

```
287  10    1:D    26     STARTINGUP,INMODULE,ININTERFACE,FLIPBYTES,
288  10    1:D    26     LIBNOTOPEN,SYSCOMP,PUBLICPROCS,GETSTMTLEV: BOOLEAN;
289  10    1:D    52
290  10    1:D    52                                          (*POINTERS:*)
291  10    1:D    52     (*INTPTR,*)REALPTR,LONGINTPTR,
292  10    1:D    52     CHARPTR,BOOLPTR,
293  10    1:D    52     TEXTPTR,NILPTR,
294  10    1:D    52     INTRACTVPTR,STRGPTR: STP;           (*POINTERS TO STANDARD IDS*)
295  10    1:D    60
296  10    1:D    60     UTYPPTR,UCSTPTR,UVARPTR,
297  10    1:D    60     UFLDPTR,UPRCPTR,UFCTPTR,            (*POINTERS TO UNDECLARED IDS*)
298  10    1:D    60     MODPTR,INPUTPTR,OUTPUTPTR,
299  10    1:D    60     OUTERBLOCK,FWPTR,USINGLIST: CTP;
300  10    1:D    72
301  10    1:D    72     GLOBTESTP: TESTP;                   (*LAST TESTPOINTER*)
302  10    1:D    73
303  10    1:D    73     LEVEL: LEVRANGE;                    (*CURRENT STATIC LEVEL*)
304  10    1:D    74     BEGSTMTLEV,STMTLEV: INTEGER;        (*CURRENT STATEMENT NESTING LEVEL*)
305  10    1:D    76     MARKP: ^INTEGER;                    (*FOR MARKING HEAP*)
306  10    1:D    77     TOS: ^LEXSTKREC;                    (*TOP OF LEX STACK*)
307  10    1:D    78     GLEV: DISPRANGE;                    (*GLOBAL LEVEL OF DISPLAY*)
308  10    1:D    79     NEWBLOCK: BOOLEAN;                  (*INDICATES NEED TO PUSH LEX STACK*)
309  10    1:D    80
310  10    1:D    80     NEXTSEG: SEGRANGE;                  (*NEXT SEGMENT #*)
311  10    1:D    81     SEGINX: INTEGER;                    (*CURRENT INDEX IN SEGMENT*)
312  10    1:D    82     SCONST: CSP;                        (*INSYMBOL STRING RESULTS*)
313  10    1:D    83     STRGCSTIC: ADDRRANGE;               (*ADDR OF LAST STRING IN CODE*)
314  10    1:D    84
315  10    1:D    84     LOWTIME,LINEINFO,SCREENDOTS,STARTDOTS,SYMBLK,SMALLESTSPACE: INTEGER;
316  10    1:D    90     LINESTART: CURSRANGE;
317  10    1:D    91
318  10    1:D    91     CURPROC,NEXTPROC: PROCRANGE;        (*PROCEDURE NUMBER ASSIGNMENT*)
319  10    1:D    93
320  10    1:D    93     CONSTBEGSYS,SIMPTYPEBEGSYS,TYPEBEGSYS,
321  10    1:D    93     BLOCKBEGSYS,SELECTSYS,FACBEGSYS,STATBEGSYS,TYPEDELS: SETOFSYS;
322  10    1:D    25     VARS: SETOFIDS;
323  10    1:D    26
324  10    1:D    26     DISPLAY: ARRAY [DISPRANGE] OF
325  10    1:D    26                 RECORD
326  10    1:D    26                   FNAME: CTP;
327  10    1:D    26                   CASE OCCUR: WHERE OF
```

```
328  10    1:D    26                    BLCK: (FFILE: CTP; FLABEL: LABELP);
329  10    1:D    26                    CREC: (CLEV: LEVRANGE; CDSPL: ADDRRANGE);
330  10    1:D    26                    VREC: (VDSPL: ADDRRANGE)
331  10    1:D    26                  END;
332  10    1:D    78
333  10    1:D    78      PFNUMOF: NONRESPFLIST;
334  10    1:D    84
335  10    1:D    84      PROCTABLE: ARRAY [PROCRANGE] OF INTEGER;
336  10    1:D    34
337  10    1:D    34      SEGTABLE: ARRAY [SEGRANGE] OF
338  10    1:D    34                  RECORD
339  10    1:D    34                    DISKADDR,CODELENG: INTEGER;
340  10    1:D    34                    SEGNAME: ALPHA;
341  10    1:D    34                    SEGKIND,
342  10    1:D    34                    TEXTADDR: INTEGER
343  10    1:D    34                  END (*SEGTABLE*) ;
344  10    1:D    62
345  10    1:D    62      COMMENT: ^STRING;
346  10    1:D    63      SYSTEMLIB: STRING[40];
347  10    1:D    84      NEXTJTAB: JTABRANGE;
348  10    1:D    85      JTAB: ARRAY [JTABRANGE] OF INTEGER;
349  10    1:D    10
350  10    1:D    10      REFFILE: FILE;
351  10    1:D    50      NREFS,REFBLK: INTEGER;
352  10    1:D    52      REFLIST: ^REFARRAY;
353  10    1:D    53      OLDSYMBLK,PREVSYMBLK: INTEGER;
354  10    1:D    55      OLDSYMCURSOR,OLDLINESTART,PREVSYMCURSOR,PREVLINESTART: CURSRANGE;
355  10    1:D    59      USEFILE: UNITFILE;
356  10    1:D    60      INCLFILE,LIBRARY: FILE;
357  10    1:D    40      LP: TEXT;
358  10    1:D    41
359  10    1:D    41      CURBYTE, CURBLK: INTEGER;
360  10    1:D    43      DISKBUF: PACKED ARRAY [0..511] OF CHAR;
361  10    1:D  1 99
362  10    1:D  1 99 (*-------------------------------------------------------------*)
363  10    1:D  1 99
364  10    1:D  1 99 (* FORWARD DECLARED PROCEDURES NEEDED BY COMPINIT *)
365  10    1:D  1 99
366  10    2:D    1 PROCEDURE ERROR(ERRORNUM: INTEGER);
367  10    2:D    2   FORWARD;
368  10    3:D    1 PROCEDURE GETNEXTPAGE;
```

```
369  10    3:D    1   FORWARD;
370  10    4:D    1 PROCEDURE PRINTLINE;
371  10    4:D    1   FORWARD;
372  10    5:D    1 PROCEDURE ENTERID(FCP: CTP);
373  10    5:D    2   FORWARD;
374  10    6:D    1 PROCEDURE INSYMBOL;
375  10    6:D    1   FORWARD;
376  10    6:D    1
377  10    6:D    1 (* FORWARD DECLARED PROCEDURES USED IN BOTH DECLARATIONPART AND BODYPART *)
378  10    6:D    1
379  10    7:D    1 PROCEDURE SEARCHSECTION(FCP:CTP; VAR FCP1: CTP);
380  10    7:D    3   FORWARD;
381  10    8:D    1 PROCEDURE SEARCHID(FIDCLS: SETOFIDS; VAR FCP: CTP);
382  10    8:D    3   FORWARD;
383  10    9:D    1 PROCEDURE GETBOUNDS(FSP: STP; VAR FMIN,FMAX: INTEGER);
384  10    9:D    4   FORWARD;
385  10   10:D    1 PROCEDURE SKIP(FSYS: SETOFSYS);
386  10   10:D    5   FORWARD;
387  10   11:D    3 FUNCTION PAOFCHAR(FSP: STP): BOOLEAN;
388  10   11:D    4   FORWARD;
389  10   12:D    3 FUNCTION STRGTYPE(FSP: STP): BOOLEAN;
390  10   12:D    4   FORWARD;
391  10   13:D    3 FUNCTION DECSIZE(I: INTEGER): INTEGER;
392  10   13:D    4   FORWARD;
393  10   14:D    1 PROCEDURE CONSTANT(FSYS: SETOFSYS; VAR FSP: STP; VAR FVALU: VALU);
394  10   14:D    7   FORWARD;
395  10   15:D    3 FUNCTION COMPTYPES(FSP1,FSP2: STP): BOOLEAN;
396  10   15:D    5   FORWARD;
397  10   16:D    1 PROCEDURE GENBYTE(FBYTE: INTEGER);
398  10   16:D    2   FORWARD;
399  10   17:D    1 PROCEDURE GENWORD(FWORD: INTEGER);
400  10   17:D    2   FORWARD;
401  10   18:D    1 PROCEDURE WRITETEXT;
402  10   18:D    1   FORWARD;
403  10   19:D    1 PROCEDURE WRITECODE(FORCEBUF: BOOLEAN);
404  10   19:D    2   FORWARD;
405  10   20:D    1 PROCEDURE BLOCK(FSYS: SETOFSYS);
406  10   20:D    5   FORWARD;
407  10   20:D    5
408  10   20:D    5 (*$I #5:COMPGLBLS.TEXT*)
408  10   20:D    5 (*$I #5:COMPINIT.TEXT*)
```

```
409  10    20:0     5
410  11     1:0     1 SEGMENT PROCEDURE COMPINIT;
411  11     1:0     1
412  11     2:0     1   PROCEDURE ENTSTDTYPES;
413  11     2:0     0   BEGIN
414  11     2:1     0     NEW(INTPTR,SCALAR,STANDARD);
415  11     2:1     5     WITH INTPTR^ DO
416  11     2:2     8       BEGIN SIZE := INTSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
417  11     2:1    21     NEW(REALPTR,SCALAR,STANDARD);
418  11     2:1    26     WITH REALPTR^ DO
419  11     2:2    30       BEGIN SIZE := REALSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
420  11     2:1    43     NEW(LONGINTPTR,LONGINT);
421  11     2:1    48     WITH LONGINTPTR^ DO
422  11     2:2    52       BEGIN SIZE := DECSIZE((BITSPERWD-1)*100 DIV 332 + 1); FORM := LONGINT END;
423  11     2:1    75     NEW(CHARPTR,SCALAR,STANDARD);
424  11     2:1    80     WITH CHARPTR^ DO
425  11     2:2    84       BEGIN SIZE := CHARSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
426  11     2:1    97     NEW(BOOLPTR,SCALAR,DECLARED);
427  11     2:1    02     WITH BOOLPTR^ DO
428  11     2:2    06       BEGIN SIZE := BOOLSIZE; FORM := SCALAR; SCALKIND := DECLARED END;
429  11     2:1    19     NEW(NILPTR,POINTER);
430  11     2:1    24     WITH NILPTR^ DO
431  11     2:2    28       BEGIN SIZE := PTRSIZE; FORM := POINTER; ELTYPE := NIL END;
432  11     2:1    41     NEW(TEXTPTR,FILES);
433  11     2:1    46     WITH TEXTPTR^ DO
434  11     2:2    50       BEGIN SIZE := FILESIZE+CHARSIZE; FORM := FILES; FILTYPE := CHARPTR END;
435  11     2:1    68     NEW(INTRACTVPTR,FILES);
436  11     2:1    73     WITH INTRACTVPTR^ DO
437  11     2:2    77       BEGIN SIZE := FILESIZE+CHARSIZE; FORM := FILES; FILTYPE := CHARPTR END;
438  11     2:1    95     NEW(STRGPTR,ARRAYS,TRUE,TRUE);
439  11     2:1    00     WITH STRGPTR^ DO
440  11     2:2    04       BEGIN FORM := ARRAYS; SIZE := (DEFSTRGLGTH + CHRSPERWD) DIV CHRSPERWD;
441  11     2:3    16         AISPACKD := TRUE; AISSTRNG := TRUE; INXTYPE := INTPTR;
442  11     2:3    31         ELWIDTH := BITSPERCHR; ELSPERWD := CHRSPERWD;
443  11     2:3    41         AELTYPE := CHARPTR; MAXLENG := DEFSTRGLGTH;
444  11     2:2    52       END
445  11     2:0    52   END (*ENTSTDTYPES*) ;
446  11     2:0    64
447  11     3:0     1   PROCEDURE ENTSTDNAMES;
448  11     3:0     1     VAR CP,CP1: CTP; I: INTEGER;
449  11     3:0     0   BEGIN
```

```
450  11    3:1     0     NEW(CP,TYPES);
451  11    3:1     5     WITH CP^ DO
452  11    3:2     8       BEGIN NAME := 'INTEGER '; IDTYPE := INTPTR; KLASS := TYPES END;
453  11    3:1    32     ENTERID(CP);
454  11    3:1    36     NEW(CP,TYPES);
455  11    3:1    41     WITH CP^ DO
456  11    3:2    44       BEGIN NAME := 'REAL    '; IDTYPE := REALPTR; KLASS := TYPES END;
457  11    3:1    69     ENTERID(CP);
458  11    3:1    73     NEW(CP,TYPES);
459  11    3:1    78     WITH CP^ DO
460  11    3:2    81       BEGIN NAME := 'CHAR    '; IDTYPE := CHARPTR; KLASS := TYPES END;
461  11    3:1    06     ENTERID(CP);
462  11    3:1    10     NEW(CP,TYPES);
463  11    3:1    15     WITH CP^ DO
464  11    3:2    18       BEGIN NAME := 'BOOLEAN '; IDTYPE := BOOLPTR; KLASS := TYPES END;
465  11    3:1    43     ENTERID(CP);
466  11    3:1    47     NEW(CP,TYPES);
467  11    3:1    52     WITH CP^ DO
468  11    3:2    55       BEGIN NAME := 'STRING  '; IDTYPE := STRGPTR; KLASS := TYPES END;
469  11    3:1    80     ENTERID(CP);
470  11    3:1    84     NEW(CP,TYPES);
471  11    3:1    89     WITH CP^ DO
472  11    3:2    92       BEGIN NAME := 'TEXT    '; IDTYPE := TEXTPTR; KLASS := TYPES END;
473  11    3:1    17     ENTERID(CP);
474  11    3:1    21     NEW(CP,TYPES);
475  11    3:1    26     WITH CP^ DO
476  11    3:2    29       BEGIN NAME := 'INTERACT'; IDTYPE := INTRACTVPTR; KLASS := TYPES END;
477  11    3:1    54     ENTERID(CP);
478  11    3:1    58     NEW(INPUTPTR,FORMALVARS,FALSE);
479  11    3:1    63     WITH INPUTPTR^ DO
480  11    3:2    67       BEGIN NAME := 'INPUT   '; IDTYPE := TEXTPTR; KLASS := FORMALVARS;
481  11    3:3    92         VLEV := 0; VADDR := 2
482  11    3:2    00       END;
483  11    3:1    02     ENTERID(INPUTPTR);
484  11    3:1    07     NEW(OUTPUTPTR,FORMALVARS,FALSE);
485  11    3:1    12     WITH OUTPUTPTR^ DO
486  11    3:2    16       BEGIN NAME := 'OUTPUT  '; IDTYPE := TEXTPTR; KLASS := FORMALVARS;
487  11    3:3    41         VLEV := 0; VADDR := 3
488  11    3:2    49       END;
489  11    3:1    51     ENTERID(OUTPUTPTR);
490  11    3:1    56     NEW(CP,FORMALVARS,FALSE);
```

```
491  11    3:1    61      WITH CP^ DO
492  11    3:2    64        BEGIN NAME := 'KEYBOARD'; IDTYPE := TEXTPTR; KLASS := FORMALVARS;
493  11    3:3    89          VLEV := 0; VADDR := 4
494  11    3:2    97        END;
495  11    3:1    99      ENTERID(CP);
496  11    3:1    03      CP1 := NIL;
497  11    3:1    06      FOR I := 0 TO 1 DO
498  11    3:2    17        BEGIN NEW(CP,KONST);
499  11    3:3    22          WITH CP^ DO
500  11    3:4    25            BEGIN IDTYPE := BOOLPTR;
501  11    3:5    31              IF I = 0 THEN NAME := 'FALSE    '
502  11    3:5    37              ELSE NAME := 'TRUE     ';
503  11    3:5    66              NEXT := CP1; VALUES.IVAL := I; KLASS := KONST
504  11    3:4    79            END;
505  11    3:3    81          ENTERID(CP); CP1 := CP
506  11    3:2    85        END;
507  11    3:1    95      BOOLPTR^.FCONST := CP;
508  11    3:1    01      NEW(CP,KONST);
509  11    3:1    06      WITH CP^ DO
510  11    3:2    09        BEGIN NAME := 'NIL      '; IDTYPE := NILPTR;
511  11    3:3    29          NEXT := NIL; VALUES.IVAL := 0; KLASS := KONST
512  11    3:2    42        END;
513  11    3:1    44      ENTERID(CP);
514  11    3:1    48      NEW(CP,KONST);
515  11    3:1    53      WITH CP^ DO
516  11    3:2    56        BEGIN
517  11    3:3    56          NAME := 'MAXINT  '; IDTYPE := INTPTR;
518  11    3:3    75          KLASS := KONST; VALUES.IVAL := MAXINT
519  11    3:2    83        END;
520  11    3:1    87      ENTERID(CP);
521  11    3:0    91    END (*ENTSTDNAMES*) ;
522  11    3:0    06
523  11    4:D     1    PROCEDURE ENTUNDECL;
524  11    4:0     0    BEGIN
525  11    4:1     0      NEW(UTYPPTR,TYPES);
526  11    4:1     5      WITH UTYPPTR^ DO
527  11    4:2     9        BEGIN NAME := '         '; IDTYPE := NIL; KLASS := TYPES END;
528  11    4:1    33      NEW(UCSTPTR,KONST);
529  11    4:1    38      WITH UCSTPTR^ DO
530  11    4:2    42        BEGIN NAME := '         '; IDTYPE := NIL; NEXT := NIL;
531  11    4:3    66          VALUES.IVAL := 0; KLASS := KONST
```

```
532  11    4:2    74      END;
533  11    4:1    76    NEW(UVARPTR,ACTUALVARS,FALSE);
534  11    4:1    81    WITH UVARPTR^ DO
535  11    4:2    85      BEGIN NAME := '          '; IDTYPE := NIL;
536  11    4:3    04        NEXT := NIL; VLEV := 0; VADDR := 0; KLASS := ACTUALVARS
537  11    4:2    22      END;
538  11    4:1    24    NEW(UFLDPTR,FIELD);
539  11    4:1    29    WITH UFLDPTR^ DO
540  11    4:2    33      BEGIN NAME := '          '; IDTYPE := NIL; NEXT := NIL;
541  11    4:3    57        FLDADDR := 0; KLASS := FIELD
542  11    4:2    65      END;
543  11    4:1    67    NEW(UPRCPTR,PROC,DECLARED,ACTUAL,FALSE);
544  11    4:1    72    WITH UPRCPTR^ DO
545  11    4:2    76      BEGIN NAME := '          '; IDTYPE := NIL; FORWDECL := FALSE;
546  11    4:3    00        NEXT := NIL; INSCOPE := FALSE; LOCALLC := 0; EXTURNAL := FALSE;
547  11    4:3    20        PFLEV := 0; PFNAME := 0; PFSEG := 0;
548  11    4:3    35        KLASS := PROC; PFDECKIND := DECLARED; PFKIND := ACTUAL
549  11    4:2    48      END;
550  11    4:1    50    NEW(UFCTPTR,FUNC,DECLARED,ACTUAL,FALSE);
551  11    4:1    55    WITH UFCTPTR^ DO
552  11    4:2    59      BEGIN NAME := '          '; IDTYPE := NIL; NEXT := NIL;
553  11    4:3    83        FORWDECL := FALSE; EXTURNAL := FALSE; INSCOPE := FALSE; LOCALLC := 0;
554  11    4:3    03        PFLEV := 0; PFNAME := 0; PFSEG := 0;
555  11    4:3    18        KLASS := FUNC; PFDECKIND := DECLARED; PFKIND := ACTUAL
556  11    4:2    31      END
557  11    4:0    33  END (*ENTUNDECL*) ;
558  11    4:0    46
559  11    5:D     1  PROCEDURE ENTSPCPROCS;
560  11    5:D     1    LABEL 1;
561  11    5:D     1    VAR LCP: CTP; I: INTEGER; ISFUNC: BOOLEAN;
562  11    5:D     4        NA: ARRAY [1..43] OF ALPHA;
563  11    5:0     0  BEGIN
564  11    5:1     0    NA[ 1] := 'READ    '; NA[ 2] := 'READLN  '; NA[ 3] := 'WRITE   ';
565  11    5:1    60    NA[ 4] := 'WRITELN '; NA[ 5] := 'EOF     '; NA[ 6] := 'EOLN    ';
566  11    5:1    20    NA[ 7] := 'PRED    '; NA[ 8] := 'SUCC    '; NA[ 9] := 'ORD     ';
567  11    5:1    80    NA[10] := 'SQR     '; NA[11] := 'ABS     '; NA[12] := 'NEW     ';
568  11    5:1    40    NA[13] := 'UNITREAD'; NA[14] := 'UNITWRIT'; NA[15] := 'CONCAT  ';
569  11    5:1    00    NA[16] := 'LENGTH  '; NA[17] := 'INSERT  '; NA[18] := 'DELETE  ';
570  11    5:1    60    NA[19] := 'COPY    '; NA[20] := 'POS     '; NA[21] := 'MOVELEFT';
571  11    5:1    20    NA[22] := 'MOVERIGH'; NA[23] := 'EXIT    '; NA[24] := 'IDSEARCH';
572  11    5:1    80    NA[25] := 'TREESEAR'; NA[26] := 'TIME    '; NA[27] := 'FILLCHAR';
```

```
573  11    5:1    40      NAC28] := 'OPENNEW '; NAC29] := 'OPENOLD '; NAC30] := 'REWRITE ';
574  11    5:1    00      NAC31] := 'CLOSE   '; NAC32] := 'SEEK    '; NAC33] := 'RESET   ';
575  11    5:1    60      NAC34] := 'GET     '; NAC35] := 'PUT     '; NAC36] := 'SCAN    ';
576  11    5:1    20      NAC37] := 'BLOCKREA'; NAC38] := 'BLOCKWRI'; NAC39] := 'TRUNC   ';
577  11    5:1    80      NAC40] := 'PAGE    '; NAC41] := 'SIZEOF  '; NAC42] := 'STR     ';
578  11    5:1    40      NAC43] := 'GOTOXY  ';
579  11    5:1    60      FOR I := 1 TO 43 DO
580  11    5:2    74        BEGIN
581  11    5:3    74          IF TINY THEN
582  11    5:4    78              IF I IN C2,7,8,10,13,17,18,19,20,32,34,35,40,42,43] THEN
583  11    5:5    92                GOTO 1;
584  11    5:3    94          ISFUNC := I IN C5,6,7,8,9,10,11,15,16,19,20,25,36,37,38,39,41];
585  11    5:3    08          IF ISFUNC THEN NEW(LCP,FUNC,SPECIAL)
586  11    5:3    16          ELSE NEW(LCP,PROC,SPECIAL);
587  11    5:3    23          WITH LCP^ DO
588  11    5:4    27            BEGIN NAME := NAC I]; NEXT := NIL; IDTYPE := NIL;
589  11    5:5    53              IF ISFUNC THEN KLASS := FUNC ELSE KLASS := PROC;
590  11    5:5    72              PFDECKIND := SPECIAL; KEY := I
591  11    5:4    84            END;
592  11    5:3    86          ENTERID(LCP);
593  11    5:3    90 1:     END
594  11    5:0    90      END (*ENTSPCPROCS*) ;
595  11    5:0  1 12
596  11    6:D    1     PROCEDURE ENTSTDPROCS;
597  11    6:D    1       VAR LCP,PARAM: CTP; LSP,FTYPE: STP; I: INTEGER; ISPROC: BOOLEAN;
598  11    6:D    7           NA: ARRAY C1,.19] OF ALPHA;
599  11    6:0    0     BEGIN
600  11    6:1    0       NAC 1] := 'ODD     '; NAC 2] := 'CHR     '; NAC 3] := 'MEMAVAIL';
601  11    6:1    60      NAC 4] := 'ROUND   '; NAC 5] := 'SIN     '; NAC 6] := 'COS     ';
602  11    6:1    20      NAC 7] := 'LOG     '; NAC 8] := 'ATAN    '; NAC 9] := 'LN      ';
603  11    6:1    80      NAC10] := 'EXP     '; NAC11] := 'SQRT    '; NAC12] := 'MARK    ';
604  11    6:1    40      NAC13] := 'RELEASE '; NAC14] := 'IORESULT'; NAC15] := 'UNITBUSY';
605  11    6:1    00      NAC16] := 'PWROFTEN'; NAC17] := 'UNITWAIT'; NAC18] := 'UNITCLEA';
606  11    6:1    60      NAC19] := 'HALT    ';
607  11    6:1    80      FOR I := 1 TO 19 DO
608  11    6:2    92        BEGIN ISPROC := I IN C12,13,17,18,19];
609  11    6:3    04          CASE I OF
610  11    6:3    07            1:  BEGIN FTYPE := BOOLPTR; NEW(PARAM,ACTUALVARS,FALSE);
611  11    6:5    16                  WITH PARAM^ DO
612  11    6:6    19                    BEGIN IDTYPE := INTPTR; KLASS := ACTUALVARS END
613  11    6:4    31                END;
```

```
614  11    6:3    33            2:  FTYPE := CHARPTR;
615  11    6:3    39            3:  BEGIN FTYPE := INTPTR; PARAM := NIL END;
616  11    6:3    47            4:  BEGIN FTYPE := INTPTR; NEW(PARAM,ACTUALVARS,FALSE);
617  11    6:5    55                  WITH PARAM^ DO BEGIN IDTYPE := REALPTR; KLASS := ACTUALVARS END
618  11    6:4    71                END;
619  11    6:3    73            5:  FTYPE := REALPTR;
620  11    6:3    79           12:  BEGIN FTYPE := NIL; NEW(PARAM,FORMALVARS,FALSE); NEW(LSP,POINTER);
621  11    6:5    92                  WITH LSP^ DO
622  11    6:6    95                    BEGIN SIZE := PTRSIZE; FORM := POINTER; ELTYPE := NIL END;
623  11    6:5    11                  WITH PARAM^ DO BEGIN IDTYPE := LSP; KLASS := FORMALVARS END
624  11    6:4    26                END;
625  11    6:3    28           14:  BEGIN FTYPE := INTPTR; PARAM := NIL END;
626  11    6:3    36           15:  BEGIN FTYPE := BOOLPTR; NEW(PARAM,ACTUALVARS,FALSE);
627  11    6:5    45                WITH PARAM^ DO
628  11    6:6    48                  BEGIN IDTYPE := INTPTR; KLASS := ACTUALVARS END;
629  11    6:4    60                END;
630  11    6:3    62           16:  FTYPE := REALPTR;
631  11    6:3    68           17:  FTYPE := NIL;
632  11    6:3    73           19:  BEGIN FTYPE := NIL; PARAM := NIL END
633  11    6:3    79          END (*PARAM AND TYPE CASES*) ;
634  11    6:3    26          IF ISPROC THEN NEW(LCP,PROC,STANDARD)
635  11    6:3    34          ELSE NEW(LCP,FUNC,STANDARD);
636  11    6:3    41          WITH LCP^ DO
637  11    6:4    44            BEGIN NAME := NAME[I]; PFDECKIND := STANDARD; CSPNUM := I + 20;
638  11    6:5    69              IF ISPROC THEN KLASS := PROC ELSE KLASS := FUNC;
639  11    6:5    86              IF PARAM <> NIL THEN PARAM^.NEXT := NIL;
640  11    6:5    96              IDTYPE := FTYPE; NEXT := PARAM
641  11    6:4    06            END;
642  11    6:3    08          ENTERID(LCP)
643  11    6:2    09        END
644  11    6:0    12    END (*ENTSTDPROCS*) ;
645  11    6:0    40
646  11    7:D     1  PROCEDURE INITSCALARS;
647  11    7:D     1   VAR I: NONRESIDENT;
648  11    7:0     0  BEGIN
649  11    7:1     0    IF MEMAVAIL > 9950 (* EMPIRICAL VALUE FOR A 50K BYTE MACHINE *) THEN
650  11    7:2     8      NOSWAP := TRUE ELSE NOSWAP := FALSE;
651  11    7:1    16    FWPTR := NIL; MODPTR := NIL; GLOBTESTP := NIL;
652  11    7:1    25    LINESTART := 0; LINEINFO := LCAFTERMARKSTACK; LIST := FALSE;
653  11    7:1    34    SYMBLK := 2; SCREENDOTS := 0; STARTDOTS := 0;
654  11    7:1    43    FOR SEG := 0 TO MAXSEG DO
```

```
655  11    7:2    54        WITH SEGTABLE[SEG] DO
656  11    7:3    62          BEGIN DISKADDR := 0; CODELENG := 0; SEGNAME := '        ';
657  11    7:4    86            SEGKIND := 0; TEXTADDR := 0
658  11    7:3    94          END;
659  11    7:1    03      USINGLIST := NIL;
660  11    7:1    06      IF USERINFO.STUPID THEN SYSTEMLIB := '*SYSTEM.PASCAL'
661  11    7:1    13      ELSE SYSTEMLIB := '*SYSTEM.LIBRARY';
662  11    7:1    57      LC := LCAFTERMARKSTACK; IOCHECK := TRUE; DP := TRUE;
663  11    7:1    66      SEGINX := 0; NEXTJTAB := 1; NEXTPROC := 2; CURPROC := 1;
664  11    7:1    79      NEW(SCONST); NEW(SYMBUFP); NEW(CODEP);
665  11    7:1    00      CLINKERINFO := FALSE; DLINKERINFO := FALSE;
666  11    7:1    06      SEG := 1; NEXTSEG := 10; CURBLK := 1; CURBYTE := 0; LSEPPROC := FALSE;
667  11    7:1    23      STARTINGUP := TRUE; NOISY := NOT USERINFO.SLOWTERM; SEPPROC := FALSE;
668  11    7:1    34      DEBUGGING := FALSE; BPTONLINE := FALSE; INMODULE := FALSE;
669  11    7:1    43      GOTOOK := FALSE; RANGECHECK := TRUE; SYSCOMP := FALSE; TINY := FALSE;
670  11    7:1    55      CODEINSEG := FALSE; PRTERR := TRUE; INCLUDING := FALSE; USING := FALSE;
671  11    7:1    67      FOR I := SEEK TO DECOPS DO PFNUMOF[I] := 0;
672  11    7:1    93      COMMENT := NIL; LIBNOTOPEN := TRUE;
673  11    7:1    00      GETSTMTLEV := TRUE; BEGSTMTLEV := 0;
674  11    7:1    06      FLIPBYTES := FALSE
675  11    7:0    06    END (*INITSCALARS*) ;
676  11    7:0    26
677  11    8:D     1    PROCEDURE INITSETS;
678  11    8:0     0    BEGIN
679  11    8:1     0      CONSTBEGSYS := [ADDOP,INTCONST,REALCONST,STRINGCONST,IDENT];
680  11    8:1    15      SIMPTYPEBEGSYS := [LPARENT] + CONSTBEGSYS;
681  11    8:1    29      TYPEBEGSYS := [ARROW,PACKEDSY,ARRAYSY,RECORDSY,SETSY,FILESY]
682  11    8:1    31                    + SIMPTYPEBEGSYS;
683  11    8:1    51      TYPEDELS := [ARRAYSY,RECORDSY,SETSY,FILESY];
684  11    8:1    67      BLOCKBEGSYS := [USESSY,LABELSY,CONSTSY,TYPESY,VARSY,
685  11    8:1    69                      PROCSY,FUNCSY,PROGSY,BEGINSY];
686  11    8:1    85      SELECTSYS := [ARROW,PERIOD,LBRACK];
687  11    8:1    99      FACBEGSYS := [INTCONST,REALCONST,LONGCONST,STRINGCONST,IDENT,
688  11    8:1    01                    LPARENT,LBRACK,NOTSY];
689  11    8:1    17      STATBEGSYS := [BEGINSY,GOTOSY,IFSY,WHILESY,REPEATSY,FORSY,WITHSY,CASESY];
690  11    8:1    31      VARS := [FORMALVARS,ACTUALVARS]
691  11    8:0    31    END (*INITSETS*) ;
692  11    8:0    50
693  11    1:0     0  BEGIN (*COMPINIT*)
694  11    1:1     0    INITSCALARS; INITSETS;
695  11    1:1     4    LEVEL := 0; TOP := 0;
```

```
696  11    1:1    10   IF NOISY THEN
697  11    1:2    14     BEGIN
698  11    1:3    14       FOR IC := 1 TO 7 DO WRITELN(OUTPUT);
699  11    1:3    38       WRITELN(OUTPUT,'PASCAL COMPILER [II.0.A.1]');
700  11    1:3    80       WRITE(OUTPUT,'<   0>')
701  11    1:2    96     END;
702  11    1:1    96   WITH DISPLAY[0] DO
703  11    1:2    03     BEGIN FNAME := NIL; FFILE := NIL; FLABEL := NIL; OCCUR := BLCK END;
704  11    1:1    21   SMALLESTSPACE:=MEMAVAIL;
705  11    1:1    25   GETNEXTPAGE;
706  11    1:1    28   INSYMBOL;
707  11    1:1    31   ENTSTDTYPES;   ENTSTDNAMES;   ENTUNDECL;
708  11    1:1    37   ENTSPCPROCS;   ENTSTDPROCS;
709  11    1:1    41   IF SYSCOMP THEN
710  11    1:2    45     BEGIN OUTERBLOCK := NIL; SEG := 0; NEXTSEG := 1;
711  11    1:3    54       GLEV :=1; BLOCKBEGSYS := BLOCKBEGSYS + [UNITSY,SEPARATSY]
712  11    1:2    64     END
713  11    1:1    80   ELSE
714  11    1:2    82     BEGIN TOP := 1; LEVEL := 1;
715  11    1:3    88       WITH DISPLAY[1] DO
716  11    1:4    95         BEGIN FNAME := NIL; FFILE := NIL;
717  11    1:5    03           FLABEL := NIL; OCCUR := BLCK
718  11    1:4    11         END;
719  11    1:3    13       LC := LC+2; GLEV := 3; (*KEEP STACK STRAIGHT FOR NOW*)
720  11    1:3    21       NEW(OUTERBLOCK,PROC,DECLARED,ACTUAL,FALSE);
721  11    1:3    26       WITH OUTERBLOCK^ DO
722  11    1:4    30         BEGIN NEXT := NIL; LOCALLC := LC;
723  11    1:5    40           NAME := 'PROGRAM '; IDTYPE := NIL; KLASS := PROC;
724  11    1:5    64           PFDECKIND := DECLARED; PFLEV := 0; PFNAME := 1; PFSEG := SEG;
725  11    1:5    84           PFKIND := ACTUAL; FORWDECL := FALSE; EXTURNAL := FALSE;
726  11    1:5    99           INSCOPE := TRUE
727  11    1:4    02         END
728  11    1:2    04     END;
729  11    1:1    04   IF SY = PROGSY THEN
730  11    1:2    09     BEGIN INSYMBOL;
731  11    1:3    12       IF SY = IDENT THEN
732  11    1:4    17         BEGIN SEGTABLE[SEG].SEGNAME := ID;
733  11    1:5    29           IF OUTERBLOCK <> NIL THEN
734  11    1:6    35             BEGIN
735  11    1:7    35               OUTERBLOCK^.NAME := ID;
736  11    1:7    41               ENTERID(OUTERBLOCK) (*ALLOWS EXIT ON PROGRAM NAME*)
```

```
737  11    1:6    43              END
738  11    1:4    46            END
739  11    1:3    46          ELSE ERROR(2); INSYMBOL;
740  11    1:3    55          IF SY = LPARENT THEN
741  11    1:4    60            BEGIN
742  11    1:5    60              REPEAT INSYMBOL
743  11    1:5    60              UNTIL SY IN [RPARENT,SEMICOLON]+BLOCKBEGSYS;
744  11    1:5    75              IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
745  11    1:4    86            END;
746  11    1:3    89          IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
747  11    1:2    00        END;
748  11    1:1    03      MARK(MARKP);
749  11    1:1    07      NEW(TOS);
750  11    1:1    12      WITH TOS^ DO  (*MAKE LEXSTKREC FOR OUTERBLOCK*)
751  11    1:2    16        BEGIN
752  11    1:3    16          PREVLEXSTACKP:=NIL;
753  11    1:3    21          BFSY:=PERIOD;
754  11    1:3    26          DFPROCP:=OUTERBLOCK;
755  11    1:3    32          DLLC:=LC;
756  11    1:3    37          DOLDLEV:=LEVEL;
757  11    1:3    43          DOLDTOP:=TOP;
758  11    1:3    46          POLDPROC:=CURPROC;
759  11    1:3    52          ISSEGMENT:=FALSE;
760  11    1:3    57          DMARKP:=MARKP;
761  11    1:2    63        END;
762  11    1:0    63  END (*COMPINIT*) ;
763  11    1:0    80  (*$I #5:COMPINIT.TEXT*)
763  11    1:0    80  (*$I #5:DECPART.A.TEXT*)
764  11    1:0    80
765  11    1:0    80  (*    COPYRIGHT (C) 1979, REGENTS OF THE         *)
766  11    1:0    80  (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO        *)
767  11    1:0    80
768  12    1:D     1  SEGMENT PROCEDURE DECLARATIONPART(FSYS: SETOFSYS);
769  12    1:D     5  VAR LSY: SYMBOL;
770  12    1:D     6      NOTDONE: BOOLEAN;
771  12    1:D     7      DUMMYVAR: ARRAY[0..0] OF INTEGER; (*FOR PRETTY DISPLAY OF STACK AND HEAP *)
772  12    1:D     8
773  12    2:D     1    PROCEDURE TYP(FSYS: SETOFSYS; VAR FSP: STP; VAR FSIZE: ADDRRANGE);
774  12    2:D     7      VAR LSP,LSP1,LSP2: STP; OLDTOP: DISPRANGE; LCP: CTP;
775  12    2:D    12          LSIZE,DISPL: ADDRRANGE; LMIN,LMAX: INTEGER;
776  12    2:D    16          PACKING: BOOLEAN; NEXTBIT,NUMBITS: BITRANGE;
```

```
777  12    2:0    19
778  12    3:0     1    PROCEDURE SIMPLETYPE(FSYS:SETOFSYS; VAR FSP:STP; VAR FSIZE:ADDRRANGE);
779  12    3:0     7      VAR LSP,LSP1: STP; LCP,LCP1: CTP; TTOP: DISPRANGE;
780  12    3:0    12          LCNT: INTEGER; LVALU: VALU;
781  12    3:0     0    BEGIN FSIZE := 1;
782  12    3:1     3      IF NOT (SY IN SIMPTYPEBEGSYS) THEN
783  12    3:2    13        BEGIN ERROR(1); SKIP(FSYS + SIMPTYPEBEGSYS) END;
784  12    3:1    33      IF SY IN SIMPTYPEBEGSYS THEN
785  12    3:2    42        BEGIN
786  12    3:3    42          IF SY = LPARENT THEN
787  12    3:4    47            BEGIN TTOP := TOP;
788  12    3:5    50              WHILE DISPLAY[TOP].OCCUR <> BLCK DO TOP := TOP - 1;
789  12    3:5    67              NEW(LSP,SCALAR,DECLARED);
790  12    3:5    72              WITH LSP^ DO
791  12    3:6    75                BEGIN SIZE := INTSIZE; FORM := SCALAR;
792  12    3:7    83                  SCALKIND := DECLARED
793  12    3:6    86                END;
794  12    3:5    88              LCP1 := NIL; LCNT := 0;
795  12    3:5    94              REPEAT INSYMBOL;
796  12    3:6    97                IF SY = IDENT THEN
797  12    3:7    02                  BEGIN NEW(LCP,KONST);
798  12    3:8    07                    WITH LCP^ DO
799  12    3:9    10                      BEGIN NAME := ID; IDTYPE := LSP; NEXT := LCP1;
800  12    3:0    25                        VALUES.IVAL := LCNT; KLASS := KONST
801  12    3:9    33                      END;
802  12    3:8    35                    ENTERID(LCP);
803  12    3:8    39                    LCNT := LCNT + 1;
804  12    3:8    44                    LCP1 := LCP; INSYMBOL
805  12    3:7    47                  END
806  12    3:6    50                ELSE ERROR(2);
807  12    3:6    56                IF NOT (SY IN FSYS + [COMMA,RPARENT]) THEN
808  12    3:7    69                  BEGIN ERROR(6); SKIP(FSYS + [COMMA,RPARENT]) END
809  12    3:5    86              UNTIL SY <> COMMA;
810  12    3:5    91              LSP^.FCONST := LCP1; TOP := TTOP;
811  12    3:5    99              IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
812  12    3:4    10            END
813  12    3:3    13          ELSE
814  12    3:4    15            BEGIN
815  12    3:5    15              IF SY = IDENT THEN
816  12    3:6    20                BEGIN SEARCHID([TYPES,KONST],LCP);
817  12    3:7    29                  INSYMBOL;
```

```
818  12    3:7    32
819  12    3:8    39
820  12    3:9    44
821  12    3:0    50
822  12    3:1    61
823  12    3:2    70
824  12    3:1    81
825  12    3:0    90
826  12    3:9    92
827  12    3:9    06
828  12    3:9    20
829  12    3:9    27
830  12    3:8    34
831  12    3:7    37
832  12    3:8    39
833  12    3:9    43
834  12    3:0    53
835  12    3:1    56
836  12    3:1    79
837  12    3:2    84
838  12    3:3    84
839  12    3:3    87
840  12    3:4    95
841  12    3:5    01
842  12    3:4    01
843  12    3:3    04
844  12    3:4    09
845  12    3:5    14
846  12    3:5    19
847  12    3:6    22
848  12    3:7    27
849  12    3:6    31
850  12    3:4    34
851  12    3:2    34
852  12    3:1    34
853  12    3:1    40
854  12    3:0    51
855  12    3:9    54
856  12    3:0    56
857  12    3:1    61
858  12    3:2    66
```

```
IF LCP^.KLASS = KONST THEN
  BEGIN NEW(LSP,SUBRANGE);
    WITH LSP^, LCP^ DO
      BEGIN RANGETYPE := IDTYPE; FORM := SUBRANGE;
        IF STRGTYPE(RANGETYPE) THEN
          BEGIN ERROR(148); RANGETYPE := NIL END;
        MIN := VALUES; SIZE := INTSIZE
      END;
    IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
    CONSTANT(FSYS,LSP1,LVALU);
    LSP^.MAX := LVALU;
    IF LSP^.RANGETYPE <> LSP1 THEN ERROR(107)
  END
ELSE
  BEGIN LSP := LCP^.IDTYPE;
    IF (LSP = STRGPTR) AND (SY = LBRACK) THEN
      BEGIN INSYMBOL;
        CONSTANT(FSYS + [RBRACK],LSP1,LVALU);
        IF LSP1 = INTPTR THEN
          BEGIN
            IF (LVALU.IVAL <= 0) OR
               (LVALU.IVAL > STRGLGTH) THEN
              BEGIN ERROR(203);
                LVALU.IVAL := DEFSTRGLGTH
              END;
            IF LVALU.IVAL <> DEFSTRGLGTH THEN
              BEGIN NEW(LSP,ARRAYS,TRUE,TRUE);
                LSP^ := STRGPTR^;
                WITH LSP^,LVALU DO
                  BEGIN MAXLENG := IVAL;
                    SIZE := (IVAL+CHRSPERWD) DIV CHRSPERWD
                  END
              END
          END
        ELSE ERROR(15);
        IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
      END
    ELSE
      IF LSP = INTPTR THEN
        IF SY = LBRACK THEN
          BEGIN INSYMBOL;
```

```
859  12    3:3    69                              NEW(LSP,LONGINT);
860  12    3:3    74                              LSP^ := LONGINTPTR^;
861  12    3:3    79                              CONSTANT(FSYS + [RBRACK],LSP1,LVALU);
862  12    3:3    01                              IF LSP1 = INTPTR THEN
863  12    3:4    06                                IF (LVALU.IVAL <= 0) OR
864  12    3:4    09                                   (LVALU.IVAL > MAXDEC) THEN ERROR(203)
865  12    3:4    18                                ELSE
866  12    3:5    23                                  LSP^.SIZE := DECSIZE(LVALU.IVAL)
867  12    3:3    25                              ELSE ERROR(15);
868  12    3:3    37                              IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12);
869  12    3:2    51                            END
870  12    3:1    51                          ELSE
871  12    3:2    53                            IF LSP^.FORM = FILES THEN
872  12    3:3    59                              IF INMODULE THEN
873  12    3:4    63                                IF NOT ININTERFACE THEN
874  12    3:5    68                                  ERROR(191); (*NO PRIVATE FILES*)
875  12    3:9    74                      IF LSP <> NIL THEN FSIZE := LSP^.SIZE
876  12    3:8    81                    END
877  12    3:6    83                END (*SY = IDENT*)
878  12    3:5    83              ELSE
879  12    3:6    85                BEGIN NEW(LSP,SUBRANGE); LSP^.FORM := SUBRANGE;
880  12    3:7    95                  CONSTANT(FSYS + [COLON],LSP1,LVALU);
881  12    3:7    12                  IF STRGTYPE(LSP1) THEN
882  12    3:8    20                    BEGIN ERROR(148); LSP1 := NIL END;
883  12    3:7    29                  WITH LSP^ DO
884  12    3:8    32                    BEGIN RANGETYPE:=LSP1; MIN:=LVALU; SIZE:=INTSIZE END;
885  12    3:7    47                  IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
886  12    3:7    61                  CONSTANT(FSYS,LSP1,LVALU);
887  12    3:7    75                  LSP^.MAX := LVALU;
888  12    3:7    82                  IF LSP^.RANGETYPE <> LSP1 THEN ERROR(107)
889  12    3:6    89                END;
890  12    3:5    92              IF LSP <> NIL THEN
891  12    3:6    97                WITH LSP^ DO
892  12    3:7    00                  IF FORM = SUBRANGE THEN
893  12    3:8    06                    IF RANGETYPE <> NIL THEN
894  12    3:9    12                      IF RANGETYPE = REALPTR THEN ERROR(399)
895  12    3:9    22                      ELSE
896  12    3:0    27                        IF MIN.IVAL > MAX.IVAL THEN
897  12    3:1    34                          BEGIN ERROR(102); MAX.IVAL := MIN.IVAL END
898  12    3:4    44            END;
899  12    3:3    44          FSP := LSP;
```

```
900  12    3:3    47          IF NOT (SY IN FSYS) THEN
901  12    3:4    57            BEGIN ERROR(6); SKIP(FSYS) END
902  12    3:2    71        END
903  12    3:1    71          ELSE FSP := NIL
904  12    3:0    74    END (*SIMPLETYPE*) ;
905  12    3:0    02
906  12    4:D     3    FUNCTION PACKABLE(FSP: STP): BOOLEAN;
907  12    4:D     4      VAR LMIN,LMAX: INTEGER;
908  12    4:0     0    BEGIN PACKABLE := FALSE;
909  12    4:1     3      IF (FSP <> NIL) AND PACKING THEN
910  12    4:2    12        WITH FSP^ DO
911  12    4:3    15          CASE FORM OF
912  12    4:3    19            SUBRANGE,
913  12    4:3    19            SCALAR:  IF (FSP <> INTPTR) AND (FSP <> REALPTR) THEN
914  12    4:5    29                       BEGIN GETBOUNDS(FSP,LMIN,LMAX);
915  12    4:6    37                         IF LMIN >= 0 THEN
916  12    4:7    42                           BEGIN PACKABLE := TRUE;
917  12    4:8    45                             NUMBITS := 1; LMIN := 1;
918  12    4:8    52                             WHILE LMIN < LMAX DO
919  12    4:9    57                               BEGIN LMIN := LMIN + 1;
920  12    4:0    62                                 LMIN := LMIN + LMIN - 1;
921  12    4:0    69                                 NUMBITS := NUMBITS + 1
922  12    4:9    72                               END
923  12    4:7    77                           END
924  12    4:5    79                       END;
925  12    4:3    81            POWER:   IF PACKABLE(ELSET) THEN
926  12    4:5    89                       BEGIN GETBOUNDS(ELSET,LMIN,LMAX);
927  12    4:6    98                         LMAX := LMAX + 1;
928  12    4:6    03                         IF LMAX < BITSPERWD THEN
929  12    4:7    08                           BEGIN PACKABLE := TRUE;
930  12    4:8    11                             NUMBITS := LMAX
931  12    4:7    11                           END
932  12    4:5    15                       END
933  12    4:3    15          END (* CASES *);
934  12    4:0    34    END (*PACKABLE*) ;
935  12    4:0    48
936  12    5:D     1    PROCEDURE FIELDLIST(FSYS: SETOFSYS; VAR FRECVAR: STP);
937  12    5:D     6      VAR LCP,LCP1,NXT,NXT1,LAST: CTP; LSP,LSP1,LSP2,LSP3,LSP4: STP;
938  12    5:D    16          MINSIZE,MAXSIZE,LSIZE: ADDRRANGE; LVALU: VALU;
939  12    5:D    20          MAXBIT,MINBIT: BITRANGE;
940  12    5:D    22
```

```
941  12    6:0     1      PROCEDURE ALLOCATE(FCP: CTP);
942  12    6:0     2        VAR ONBOUND: BOOLEAN;
943  12    6:0     0      BEGIN ONBOUND := FALSE;
944  12    6:1     3        WITH FCP^ DO
945  12    6:2     6          IF PACKABLE(IDTYPE) THEN
946  12    6:3    14            BEGIN
947  12    6:4    14              IF (NUMBITS + NEXTBIT) > BITSPERWD THEN
948  12    6:5    25                BEGIN DISPL := DISPL + 1; NEXTBIT := 0; ONBOUND := TRUE END;
949  12    6:4    40              FLDADDR := DISPL; FISPACKD := TRUE;
950  12    6:4    52              FLDWIDTH := NUMBITS; FLDRBIT := NEXTBIT;
951  12    6:4    66              NEXTBIT := NEXTBIT + NUMBITS
952  12    6:3    69            END
953  12    6:2    76          ELSE
954  12    6:3    78            BEGIN DISPL := DISPL + ORD(NEXTBIT > 0);
955  12    6:4    90              NEXTBIT := 0; ONBOUND := TRUE;
956  12    6:4    97              FISPACKD := FALSE; FLDADDR := DISPL;
957  12    6:4    09              IF IDTYPE <> NIL THEN
958  12    6:5    15                DISPL := DISPL + IDTYPE^.SIZE
959  12    6:3    20            END;
960  12    6:1    25        IF ONBOUND AND (LAST <> NIL) THEN
961  12    6:2    34          WITH LAST^ DO
962  12    6:3    39            IF FISPACKD THEN
963  12    6:4    44              IF FLDRBIT = 0 THEN FISPACKD := FALSE
964  12    6:4    54              ELSE
965  12    6:5    58                IF (FLDWIDTH <= 8) AND (FLDRBIT <= 8) THEN
966  12    6:6    71                  BEGIN FLDWIDTH := 8; FLDRBIT := 8 END
967  12    6:0    81      END (*ALLOCATE*) ;
968  12    6:0    94
969  12    7:0     1      PROCEDURE VARIANTLIST;
970  12    7:0     1        VAR GOTTAGNAME: BOOLEAN;
971  12    7:0     0      BEGIN NEW(LSP,TAGFLD);
972  12    7:1     6        WITH LSP^ DO
973  12    7:2    11          BEGIN TAGFIELDP := NIL; FSTVAR := NIL; FORM := TAGFLD END;
974  12    7:1    26        FRECVAR := LSP;
975  12    7:1    33        INSYMBOL;
976  12    7:1    36        IF SY = IDENT THEN
977  12    7:2    41          BEGIN
978  12    7:3    41            IF PACKING THEN NEW(LCP,FIELD,TRUE)
979  12    7:3    52            ELSE NEW(LCP,FIELD,FALSE);
980  12    7:3    60            WITH LCP^ DO
981  12    7:4    65              BEGIN IDTYPE := NIL; KLASS:=FIELD;
```

```
 982  12    7:5    75            NEXT := NIL; FISPACKD := FALSE
 983  12    7:4    83          END;
 984  12    7:3    85        GOTTAGNAME := FALSE; PRTERR := FALSE;
 985  12    7:3    91        SEARCHID([TYPES],LCP1); PRTERR := TRUE;
 986  12    7:3    04        IF LCP1 = NIL THEN
 987  12    7:4    11          BEGIN GOTTAGNAME := TRUE;
 988  12    7:5    14            LCP^.NAME := ID; ENTERID(LCP); INSYMBOL;
 989  12    7:5    30            IF SY = COLON THEN INSYMBOL ELSE ERROR(5)
 990  12    7:4    41          END;
 991  12    7:3    44        IF SY = IDENT THEN
 992  12    7:4    49          BEGIN SEARCHID([TYPES],LCP1);
 993  12    7:5    59            LSP1 := LCP1^.IDTYPE;
 994  12    7:5    66            IF LSP1 <> NIL THEN
 995  12    7:6    73              BEGIN
 996  12    7:7    73                IF LSP1^.FORM <= SUBRANGE THEN
 997  12    7:8    81                  BEGIN
 998  12    7:9    81                    IF COMPTYPES(REALPTR,LSP1) THEN ERROR(109);
 999  12    7:9    97                    LCP^.IDTYPE := LSP1; LSP^.TAGFIELDP := LCP;
1000  12    7:9    15                    IF GOTTAGNAME THEN ALLOCATE(LCP)
1001  12    7:8    21                  END
1002  12    7:7    23                ELSE ERROR(110)
1003  12    7:6    26              END;
1004  12    7:5    29            INSYMBOL
1005  12    7:4    29          END
1006  12    7:3    32        ELSE BEGIN ERROR(2); SKIP(FSYS + [OFSY,LPARENT]) END
1007  12    7:2    54    END
1008  12    7:1    54  ELSE BEGIN ERROR(2); SKIP(FSYS + [OFSY,LPARENT]) END;
1009  12    7:1    76  LSP^.SIZE := DISPL + ORD(NEXTBIT > 0);
1010  12    7:1    89  IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1011  12    7:1    03  LSP1 := NIL; MINSIZE := DISPL; MAXSIZE := DISPL;
1012  12    7:1    19  MINBIT := NEXTBIT; MAXBIT := NEXTBIT;
1013  12    7:1    31  REPEAT LSP2 := NIL;
1014  12    7:2    35    REPEAT CONSTANT(FSYS + [COMMA,COLON,LPARENT],LSP3,LVALU);
1015  12    7:3    55      IF LSP^.TAGFIELDP <> NIL THEN
1016  12    7:4    63        IF NOT COMPTYPES(LSP^.TAGFIELDP^.IDTYPE,LSP3) THEN
1017  12    7:5    79          ERROR(111);
1018  12    7:3    83      NEW(LSP3,VARIANT);
1019  12    7:3    89      WITH LSP3^ DO
1020  12    7:4    94        BEGIN NXTVAR := LSP1; SUBVAR := LSP2;
1021  12    7:5    08          VARVAL := LVALU; FORM := VARIANT
1022  12    7:4    19        END;
```

```
1023  12    7:3    21            LSP1 := LSP3; LSP2 := LSP3;
1024  12    7:3    33            TEST := SY <> COMMA;
1025  12    7:3    38            IF NOT TEST THEN INSYMBOL
1026  12    7:2    42          UNTIL TEST;
1027  12    7:2    48          IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1028  12    7:2    62          IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
1029  12    7:2    76          IF SY = RPARENT THEN LSP2 := NIL
1030  12    7:2    81          ELSE
1031  12    7:3    87            FIELDLIST(FSYS + [RPARENT,SEMICOLON],LSP2);
1032  12    7:2    03          IF DISPL > MAXSIZE THEN
1033  12    7:3    12            BEGIN MAXSIZE := DISPL; MAXBIT := NEXTBIT END
1034  12    7:2    24          ELSE
1035  12    7:3    26            IF (DISPL = MAXSIZE) AND (NEXTBIT > MAXBIT) THEN
1036  12    7:4    43              MAXBIT := NEXTBIT;
1037  12    7:2    49          WHILE LSP3 <> NIL DO
1038  12    7:3    56            BEGIN LSP4 := LSP3^.SUBVAR; LSP3^.SUBVAR := LSP2;
1039  12    7:4    72              LSP3^.SIZE := DISPL + ORD(NEXTBIT > 0);
1040  12    7:4    85              LSP3 := LSP4
1041  12    7:3    85            END;
1042  12    7:2    93          IF SY = RPARENT THEN
1043  12    7:3    98            BEGIN INSYMBOL;
1044  12    7:4    01              IF NOT (SY IN FSYS + [SEMICOLON]) THEN
1045  12    7:5    15                BEGIN ERROR(6); SKIP(FSYS + [SEMICOLON]) END
1046  12    7:3    33            END
1047  12    7:2    33          ELSE ERROR(4);
1048  12    7:2    39          TEST := SY <> SEMICOLON;
1049  12    7:2    44          IF NOT TEST THEN
1050  12    7:3    48            BEGIN INSYMBOL;
1051  12    7:4    51              DISPL := MINSIZE; NEXTBIT := MINBIT
1052  12    7:3    57            END
1053  12    7:1    63        UNTIL (TEST) OR (SY = ENDSY); (* <<<< SMF 2-28-78 *)
1054  12    7:1    70        DISPL := MAXSIZE; NEXTBIT := MAXBIT;
1055  12    7:1    82        LSP^.FSTVAR := LSP1
1056  12    7:0    87      END (*VARIANTLIST*) ;
1057  12    7:0    12
1058  12    5:0     0    BEGIN (*FIELDLIST*)
1059  12    5:1     0      NXT1 := NIL; LSP := NIL; LAST := NIL;
1060  12    5:1     9      IF NOT (SY IN [IDENT,CASESY]) THEN
1061  12    5:2    21        BEGIN ERROR(19); SKIP(FSYS + [IDENT,CASESY]) END;
1062  12    5:1    43      WHILE SY = IDENT DO
1063  12    5:2    48        BEGIN NXT := NXT1;
```

```
1064  12    5:3   51          REPEAT
1065  12    5:4   51            IF SY = IDENT THEN
1066  12    5:5   56              BEGIN
1067  12    5:6   56                IF PACKING THEN NEW(LCP,FIELD,TRUE)
1068  12    5:6   66                ELSE NEW(LCP,FIELD,FALSE);
1069  12    5:6   73                WITH LCP^ DO
1070  12    5:7   76                  BEGIN NAME := ID; IDTYPE := NIL; NEXT := NXT;
1071  12    5:8   94                    KLASS := FIELD; FISPACKD := FALSE
1072  12    5:7   04                  END;
1073  12    5:6   06                NXT := LCP;
1074  12    5:6   09                ENTERID(LCP);
1075  12    5:6   13                INSYMBOL
1076  12    5:5   13              END
1077  12    5:4   16            ELSE ERROR(2);
1078  12    5:4   22            IF NOT (SY IN [COMMA,COLON]) THEN
1079  12    5:5   29              BEGIN ERROR(6); SKIP(FSYS + [COMMA,COLON,SEMICOLON,CASESY]) END;
1080  12    5:4   51            TEST := SY <> COMMA;
1081  12    5:4   56            IF NOT TEST  THEN INSYMBOL
1082  12    5:3   60          UNTIL TEST;
1083  12    5:3   66          IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1084  12    5:3   80          TYP(FSYS + [CASESY,SEMICOLON],LSP,LSIZE);
1085  12    5:3   02          IF LSP <> NIL THEN
1086  12    5:4   07            IF LSP^.FORM = FILES THEN ERROR(108);
1087  12    5:3   17          WHILE NXT <> NXT1 DO
1088  12    5:4   22            WITH NXT^ DO
1089  12    5:5   25              BEGIN IDTYPE := LSP; ALLOCATE(NXT);
1090  12    5:6   34                IF NEXT = NXT1 THEN LAST := NXT;
1091  12    5:6   44                NXT := NEXT
1092  12    5:5   44              END;
1093  12    5:3   51          NXT1 := LCP;
1094  12    5:3   54          IF SY = SEMICOLON THEN
1095  12    5:4   59            BEGIN INSYMBOL;
1096  12    5:5   62              IF NOT (SY IN [IDENT,ENDSY,CASESY]) THEN (* <<<< SMF 2-28-78 *)
1097  12    5:6   75                BEGIN ERROR(19); SKIP(FSYS + [IDENT,CASESY]) END
1098  12    5:4   97            END
1099  12    5:2   97        END (*WHILE*);
1100  12    5:1   99      NXT := NIL;
1101  12    5:1   02      WHILE NXT1 <> NIL DO
1102  12    5:2   07        WITH NXT1^ DO
1103  12    5:3   10          BEGIN LCP := NEXT; NEXT := NXT; NXT := NXT1; NXT1 := LCP END;
1104  12    5:1   29      IF SY = CASESY THEN VARIANTLIST
```

```
1105  12    5:1    34      ELSE FRECVAR := NIL
1106  12    5:0    39    END (*FIELDLIST*) ;
1107  12    5:0    64
1108  12    8:0     1    PROCEDURE POINTERTYPE;
1109  12    8:0     0    BEGIN NEW(LSP,POINTER); FSP := LSP;
1110  12    8:1    13      WITH LSP^ DO
1111  12    8:2    18        BEGIN ELTYPE := NIL; SIZE := PTRSIZE; FORM := POINTER END;
1112  12    8:1    31      INSYMBOL;
1113  12    8:1    34      IF SY = IDENT THEN
1114  12    8:2    39        BEGIN PRTERR := FALSE;
1115  12    8:3    42          SEARCHID([TYPES],LCP); PRTERR := TRUE;
1116  12    8:3    55          IF LCP = NIL THEN   (*FORWARD REFERENCED TYPE ID*)
1117  12    8:4    62            BEGIN NEW(LCP,TYPES);
1118  12    8:5    68              WITH LCP^ DO
1119  12    8:6    73                BEGIN NAME := ID; IDTYPE := LSP;
1120  12    8:7    85                  NEXT := FWPTR; KLASS := TYPES
1121  12    8:6    94                END;
1122  12    8:5    96              FWPTR := LCP
1123  12    8:4    96            END
1124  12    8:3    01          ELSE
1125  12    8:4    03            BEGIN
1126  12    8:5    03              IF LCP^.IDTYPE <> NIL THEN
1127  12    8:6    11                IF (LCP^.IDTYPE^.FORM <> FILES) OR SYSCOMP THEN
1128  12    8:7    23                  LSP^.ELTYPE := LCP^.IDTYPE
1129  12    8:6    31                ELSE ERROR(108)
1130  12    8:4    36            END;
1131  12    8:3    39          INSYMBOL;
1132  12    8:2    42        END
1133  12    8:1    42      ELSE ERROR(2)
1134  12    8:0    45    END (*POINTERTYPE*) ;
1135  12    8:0    60
1136  12    2:0     0  BEGIN (*TYP*)
1137  12    2:1     0    PACKING := FALSE;
1138  12    2:1     3    IF NOT (SY IN TYPEBEGSYS) THEN
1139  12    2:2    13       BEGIN ERROR(10); SKIP(FSYS + TYPEBEGSYS) END;
1140  12    2:1    33    IF SY IN TYPEBEGSYS THEN
1141  12    2:2    42      BEGIN
1142  12    2:3    42        IF SY IN SIMPTYPEBEGSYS THEN SIMPLETYPE(FSYS,FSP,FSIZE)
1143  12    2:3    60        ELSE
1144  12    2:4    64  (*^*)   IF SY = ARROW THEN POINTERTYPE
1145  12    2:4    69          ELSE
```

```
1146  12    2:5    73                BEGIN
1147  12    2:6    73                  IF SY = PACKEDSY THEN
1148  12    2:7    78                    BEGIN INSYMBOL; PACKING := TRUE;
1149  12    2:8    84                      IF NOT (SY IN TYPEDELS) THEN
1150  12    2:9    94                        BEGIN ERROR(10); SKIP(FSYS + TYPEDELS) END
1151  12    2:7    14                    END;
1152  12    2:6    14  (*ARRAY*)       IF SY = ARRAYSY THEN
1153  12    2:7    19                    BEGIN INSYMBOL;
1154  12    2:8    22                      IF SY = LBRACK THEN INSYMBOL ELSE ERROR(11);
1155  12    2:8    36                      LSP1 := NIL;
1156  12    2:8    39                      REPEAT
1157  12    2:9    39                        IF PACKING THEN NEW(LSP,ARRAYS,TRUE,FALSE)
1158  12    2:9    47                        ELSE NEW(LSP,ARRAYS,FALSE);
1159  12    2:9    54                        WITH LSP^ DO
1160  12    2:0    57                          BEGIN AELTYPE := LSP1; INXTYPE := NIL;
1161  12    2:1    69                            IF PACKING THEN AISSTRNG := FALSE;
1162  12    2:1    78                            AISPACKD := FALSE;  FORM := ARRAYS
1163  12    2:0    88                          END;
1164  12    2:9    90                        LSP1 := LSP;
1165  12    2:9    93                        SIMPLETYPE(FSYS + [COMMA,RBRACK,OFSY],LSP2,LSIZE);
1166  12    2:9    14                        LSP1^.SIZE := LSIZE;
1167  12    2:9    17                        IF LSP2 <> NIL THEN
1168  12    2:0    22                          IF LSP2^.FORM <= SUBRANGE THEN
1169  12    2:1    28                            BEGIN
1170  12    2:2    28                              IF LSP2 = REALPTR THEN
1171  12    2:3    34                                BEGIN ERROR(109); LSP2 := NIL END
1172  12    2:2    41                              ELSE
1173  12    2:3    43                                IF LSP2 = INTPTR THEN
1174  12    2:4    48                                  BEGIN ERROR(149); LSP2 := NIL END;
1175  12    2:2    57                              LSP^.INXTYPE := LSP2
1176  12    2:1    60                            END
1177  12    2:0    62                          ELSE BEGIN ERROR(113); LSP2 := NIL END;
1178  12    2:9    71                        TEST := SY <> COMMA;
1179  12    2:9    76                        IF NOT TEST THEN INSYMBOL
1180  12    2:8    80                      UNTIL TEST;
1181  12    2:8    86                      IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12);
1182  12    2:8    00                      IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1183  12    2:8    14                      TYP(FSYS,LSP,LSIZE);
1184  12    2:8    27                      IF LSP <> NIL THEN
1185  12    2:9    32                        IF LSP^.FORM = FILES THEN ERROR(108);
1186  12    2:8    42                      IF PACKABLE(LSP) THEN
```

```
1187  12    2:9    49                    IF NUMBITS + NUMBITS <= BITSPERWD THEN
1188  12    2:0    58                      WITH LSP1^ DO
1189  12    2:1    61                        BEGIN AISPACKD := TRUE;
1190  12    2:2    67                          ELSPERWD := BITSPERWD DIV NUMBITS;
1191  12    2:2    76                          ELWIDTH := NUMBITS
1192  12    2:1    80                        END;
1193  12    2:8    83                  REPEAT
1194  12    2:9    83                    WITH LSP1^ DO
1195  12    2:0    86                      BEGIN LSP2 := AELTYPE; AELTYPE := LSP;
1196  12    2:1    97                        IF INXTYPE <> NIL THEN
1197  12    2:2    04                          BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
1198  12    2:3    14                            IF AISPACKD THEN
1199  12    2:4    19                              LSIZE := (LMAX-LMIN+ELSPERWD)
1200  12    2:4    26                                           DIV ELSPERWD
1201  12    2:3    26                            ELSE
1202  12    2:4    34                              LSIZE := LSIZE*(LMAX - LMIN + 1);
1203  12    2:3    43                            IF LSIZE <= 0 THEN
1204  12    2:4    48                              BEGIN ERROR(398); LSIZE := 1 END;
1205  12    2:3    57                            SIZE := LSIZE
1206  12    2:2    59                          END
1207  12    2:0    61                      END;
1208  12    2:9    61                    LSP := LSP1; LSP1 := LSP2
1209  12    2:8    64                  UNTIL LSP1 = NIL
1210  12    2:7    68                END
1211  12    2:6    72              ELSE
1212  12    2:7    74  (*RECORD*)    IF SY = RECORDSY THEN
1213  12    2:8    79                  BEGIN INSYMBOL;
1214  12    2:9    82                    OLDTOP := TOP;
1215  12    2:9    85                    IF TOP < DISPLIMIT THEN
1216  12    2:0    90                      BEGIN TOP := TOP + 1;
1217  12    2:1    95                        WITH DISPLAY[TOP] DO
1218  12    2:2    02                          BEGIN FNAME := NIL; OCCUR := REC END
1219  12    2:0    12                      END
1220  12    2:9    12                    ELSE ERROR(250);
1221  12    2:9    20                    DISPL := 0; NEXTBIT := 0;
1222  12    2:9    26                    FIELDLIST(FSYS-[SEMICOLON]+[ENDSY],LSP1);
1223  12    2:9    45                    DISPL := DISPL + ORD(NEXTBIT > 0);
1224  12    2:9    53                    NEW(LSP,RECORDS);
1225  12    2:9    58                    WITH LSP^ DO
1226  12    2:0    61                      BEGIN FSTFLD := DISPLAY[TOP].FNAME;
1227  12    2:1    72                        RECVAR := LSP1; SIZE := DISPL;
```

```
1228  12    2:1    82                        FORM := RECORDS
1229  12    2:0    86                      END;
1230  12    2:9    88                    TOP := OLDTOP;
1231  12    2:9    91                    IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
1232  12    2:8    02                  END
1233  12    2:7    05                ELSE
1234  12    2:8    07  (*SET*)         IF SY = SETSY THEN
1235  12    2:9    12                    BEGIN INSYMBOL;
1236  12    2:0    15                      IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1237  12    2:0    29                      SIMPLETYPE(FSYS,LSP1,LSIZE);
1238  12    2:0    42                      IF LSP1 <> NIL THEN
1239  12    2:1    47                        IF (LSP1^.FORM > SUBRANGE) OR
1240  12    2:1    51                                (LSP1 = INTPTR) OR (LSP1 = REALPTR) THEN
1241  12    2:2    62                          BEGIN ERROR(115); LSP1 := NIL END
1242  12    2:1    69                        ELSE
1243  12    2:2    71                          IF LSP1 = REALPTR THEN
1244  12    2:3    77                            BEGIN ERROR(114); LSP1 := NIL END;
1245  12    2:0    84                      NEW(LSP,POWER);
1246  12    2:0    89                      WITH LSP^ DO
1247  12    2:1    92                        BEGIN ELSET := LSP1; FORM := POWER;
1248  12    2:2    04                          IF LSP1 <> NIL THEN
1249  12    2:3    09                            BEGIN GETBOUNDS(LSP1,LMIN,LMAX);
1250  12    2:4    17                              SIZE := (LMAX + BITSPERWD) DIV BITSPERWD;
1251  12    2:4    25                              IF SIZE > 255 THEN
1252  12    2:5    34                                BEGIN ERROR(169); SIZE := 1 END
1253  12    2:3    44                            END
1254  12    2:2    44                          ELSE SIZE := 0
1255  12    2:1    48                        END
1256  12    2:9    50                    END
1257  12    2:8    50                  ELSE
1258  12    2:9    52  (*FILE*)          IF SY = FILESY THEN
1259  12    2:0    57                      BEGIN
1260  12    2:1    57                        IF INMODULE THEN
1261  12    2:2    61                          IF NOT ININTERFACE THEN
1262  12    2:3    66                            ERROR(191); (*NO PRIVATE FILES*)
1263  12    2:1    72                        INSYMBOL; NEW(LSP,FILES);
1264  12    2:1    80                        WITH LSP^ DO
1265  12    2:2    83                          BEGIN FORM := FILES; FILTYPE := NIL END;
1266  12    2:1    95                        IF SY = OFSY THEN
1267  12    2:2    00                          BEGIN INSYMBOL; TYP(FSYS,LSP1,LSIZE) END
1268  12    2:1    16                        ELSE LSP1 := NIL;
```

```
1269  12    2:1    21                          LSP^.FILTYPE := LSP1;
1270  12    2:1    26                          IF LSP1 <> NIL THEN
1271  12    2:2    31                            LSP^.SIZE := FILESIZE + LSP1^.SIZE
1272  12    2:1    36                          ELSE LSP^.SIZE := NILFILESIZE
1273  12    2:0    42                        END;
1274  12    2:6    44                FSP := LSP
1275  12    2:5    45              END;
1276  12    2:3    47            IF NOT (SY IN FSYS) THEN
1277  12    2:4    57              BEGIN ERROR(6); SKIP(FSYS) END
1278  12    2:2    71          END
1279  12    2:1    71        ELSE FSP := NIL;
1280  12    2:1    76        IF FSP = NIL THEN FSIZE := 1 ELSE FSIZE := FSP^.SIZE
1281  12    2:0    90      END (*TYP*) ;
1282  12    2:0    24 (*$I #5:DECPART.A.TEXT*)
1282  12    2:0    24 (*$I #5:DECPART.B.TEXT*)
1283  12    2:0    24
1284  12    2:0    24 (*    COPYRIGHT (C) 1979, REGENTS OF THE          *)
1285  12    2:0    24 (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO         *)
1286  12    2:0    24
1287  12    9:D     1      PROCEDURE USESDECLARATION(MAGIC: BOOLEAN);
1288  12    9:D     2        LABEL 1;
1289  12    9:D     2        TYPE DCREC = RECORD
1290  12    9:D     2                       DISKADDR: INTEGER;
1291  12    9:D     2                       CODELENG: INTEGER
1292  12    9:D     2                     END;
1293  12    9:D     2        VAR SEGDICT: RECORD
1294  12    9:D     2                       DANDC: ARRAY[SEGRANGE] OF DCREC;
1295  12    9:D     2                       SEGNAME: ARRAY[SEGRANGE] OF ALPHA;
1296  12    9:D     2                       SEGKIND: ARRAY[SEGRANGE] OF INTEGER;
1297  12    9:D     2                       TEXTADDR: ARRAY[SEGRANGE] OF INTEGER;
1298  12    9:D     2                       FILLER: ARRAY[0..127] OF INTEGER
1299  12    9:D     2                     END;
1300  12    9:D    58            FOUND: BOOLEAN; BEGADDR: INTEGER;
1301  12    9:D    60            LCP: CTP; LLEXSTK: LEXSTKREC; LNAME: ALPHA;
1302  12    9:D    76            LSY: SYMBOL; LOP: OPERATOR; LID: ALPHA;
1303  12    9:D    82
1304  12   10:D     1        PROCEDURE GETTEXT(VAR FOUND: BOOLEAN);
1305  12   10:D     2          VAR LCP: CTP; SEGINDEX: INTEGER;
1306  12   10:D     4
1307  12   10:0     0        BEGIN FOUND := FALSE;
1308  12   10:1     3          LCP := MODPTR;
```

```
1309  12    10:1     7          WHILE (LCP <> NIL) AND NOT FOUND DO
1310  12    10:2    16            IF LCP^.NAME = ID THEN FOUND := TRUE ELSE LCP := LCP^.NEXT;
1311  12    10:1    35          IF FOUND THEN
1312  12    10:2    39            BEGIN
1313  12    10:3    39              LSEPPROC := SEGTABLE[LCP^.SEGID].SEGKIND = 4;
1314  12    10:3    52              IF NOT LSEPPROC THEN
1315  12    10:4    57                BEGIN SEG := LCP^.SEGID; NEXTPROC := 1 END;
1316  12    10:3    65              BEGADDR := SEGTABLE[LCP^.SEGID].TEXTADDR;
1317  12    10:3    78              USEFILE := WORKCODE;
1318  12    10:2    82            END
1319  12    10:1    82          ELSE
1320  12    10:2    84            BEGIN FOUND := TRUE;
1321  12    10:3    87              IF LIBNOTOPEN THEN
1322  12    10:4    91                BEGIN RESET(LIBRARY,SYSTEMLIB);
1323  12    10:5    02                  IF IORESULT <> 0 THEN BEGIN ERROR(187); FOUND := FALSE END
1324  12    10:5    17                  ELSE
1325  12    10:6    19                    IF BLOCKREAD(LIBRARY,SEGDICT,1,0) <> 1 THEN
1326  12    10:7    38                      BEGIN ERROR(187); FOUND := FALSE END;
1327  12    10:4    47                END;
1328  12    10:3    47              IF FOUND THEN
1329  12    10:4    51                BEGIN LIBNOTOPEN := FALSE;
1330  12    10:5    54                  SEGINDEX := 0; FOUND := FALSE;
1331  12    10:5    60                  WHILE (SEGINDEX <= MAXSEG) AND (NOT FOUND) DO
1332  12    10:6    69                    IF MAGIC THEN
1333  12    10:7    74                        IF SEGDICT.SEGNAME[SEGINDEX] = LNAME THEN FOUND := TRUE
1334  12    10:7    90                        ELSE SEGINDEX := SEGINDEX + 1
1335  12    10:6    95                    ELSE
1336  12    10:7    01                      IF SEGDICT.SEGNAME[SEGINDEX] = ID THEN FOUND := TRUE
1337  12    10:7    15                      ELSE SEGINDEX := SEGINDEX + 1;
1338  12    10:5    26                  IF FOUND THEN
1339  12    10:6    30                    BEGIN USEFILE := SYSLIBRARY;
1340  12    10:7    34                      BEGADDR := SEGDICT.TEXTADDR[SEGINDEX];
1341  12    10:7    45                      LSEPPROC := SEGDICT.SEGKIND[SEGINDEX] = 4;
1342  12    10:7    56                      IF NOT LSEPPROC THEN
1343  12    10:8    61                        BEGIN
1344  12    10:9    61                          IF MAGIC THEN SEG := 6
1345  12    10:9    66                          ELSE
1346  12    10:0    71                            BEGIN SEG := NEXTSEG;
1347  12    10:1    75                              NEXTSEG := NEXTSEG + 1;
1348  12    10:1    81                              IF NEXTSEG > MAXSEG THEN ERROR(250)
1349  12    10:0    90                            END;
```

```
1350  12   10:9   93                          WITH SEGTABLE[SEG] DO
1351  12   10:0   01                            BEGIN DISKADDR := 0; CODELENG := 0;
1352  12   10:1   09                              SEGNAME := SEGDICT.SEGNAME[SEGINDEX];
1353  12   10:1   20                              IF INMODULE OR MAGIC THEN SEGKIND := 0
1354  12   10:1   31                              ELSE SEGKIND := SEGDICT.SEGKIND[SEGINDEX];
1355  12   10:1   46                              TEXTADDR := 0
1356  12   10:0   49                            END;
1357  12   10:9   51                          NEXTPROC := 1
1358  12   10:8   51                        END
1359  12   10:6   54                    END
1360  12   10:5   54                  ELSE ERROR(190) (*NOT IN LIBRARY*)
1361  12   10:4   59                END
1362  12   10:2   62            END;
1363  12   10:1   62          IF BEGADDR = 0 THEN BEGIN ERROR(195); FOUND := FALSE END;
1364  12   10:1   79          IF FOUND THEN
1365  12   10:2   83            BEGIN
1366  12   10:3   83              USING := TRUE;
1367  12   10:3   86              PREVSYMCURSOR := SYMCURSOR;
1368  12   10:3   90              PREVLINESTART := LINESTART;
1369  12   10:3   95              PREVSYMBLK := SYMBLK - 2;
1370  12   10:3   02              SYMBLK := BEGADDR; GETNEXTPAGE;
1371  12   10:3   11              INSYMBOL
1372  12   10:2   11            END
1373  12   10:0   14        END (*GETTEXT*) ;
1374  12   10:0   34
1375  12    9:0    0      BEGIN (*USESDECLARATION*)
1376  12    9:1    0        IF LEVEL <> 1 THEN ERROR(189);
1377  12    9:1   12        IF INMODULE AND NOT ININTERFACE THEN ERROR(192);
1378  12    9:1   26        IF NOT MAGIC THEN DLINKERINFO := TRUE;
1379  12    9:1   33        IF NOT USING THEN USINGLIST := NIL;
1380  12    9:1   41        REPEAT
1381  12    9:2   41          IF (NOT MAGIC) AND (SY <> IDENT) THEN ERROR(2)
1382  12    9:2   50          ELSE
1383  12    9:3   55            IF USING THEN
1384  12    9:4   59              BEGIN LCP := USINGLIST;
1385  12    9:5   64                WHILE LCP <> NIL DO
1386  12    9:6   71                  IF LCP^.NAME = ID THEN GOTO 1
1387  12    9:6   83                  ELSE LCP := LCP^.NEXT;
1388  12    9:5   94                ERROR(188)(*UNIT MUST BE PREDECLARED IN MAIN PROG*);
1389  12    9:5   00            1:
1390  12    9:4   00              END
```

```
1391  12    9:3    00          ELSE
1392  12    9:4    02            BEGIN
1393  12    9:5    J2              IF MAGIC THEN
1394  12    9:6    05                BEGIN LNAME := 'TURTLE  ';
1395  12    9:7    21                  LSY := SY; LOP := OP; LID := ID
1396  12    9:6    33                END
1397  12    9:5    37              ELSE
1398  12    9:6    39                BEGIN LNAME := ID;
1399  12    9:7    46                  WRITELN(OUTPUT); WRITELN(OUTPUT,ID,' [',MEMAVAIL:5,' WORDS]');
1400  12    9:7    06                  WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
1401  12    9:6    31                END;
1402  12    9:5    31              WITH LLEXSTK DO
1403  12    9:6    31                BEGIN DOLDSEG := SEG; SOLDPROC := NEXTPROC END;
1404  12    9:5    40              GETTEXT(FOUND);
1405  12    9:5    45              IF FOUND THEN
1406  12    9:6    50                BEGIN
1407  12    9:7    50                  NEW(LCP,MODULE);
1408  12    9:7    56                  WITH LCP^ DO
1409  12    9:8    62                    BEGIN NAME := LNAME; NEXT := USINGLIST;
1410  12    9:9    78                      IDTYPE := NIL; KLASS := MODULE;
1411  12    9:9    92                      IF LSEPPROC THEN SEGID := -1 (*NO SEG*) ELSE SEGID := SEG
1412  12    9:8    11                    END;
1413  12    9:7    13                  ENTERID(LCP);
1414  12    9:7    19                  USINGLIST := LCP;
1415  12    9:7    24                  DECLARATIONPART(FSYS + [ENDSY]);
1416  12    9:7    39                  IF NEXTPROC=1 (*NO PROCS DECLARED*) THEN
1417  12    9:8    45                    LCP^.SEGID := -1; (*NO SEG*)
1418  12    9:7    53                  SYMBLK := 9999; (*FORCE RETURN TO SOURCEFILE*)
1419  12    9:7    58                  GETNEXTPAGE
1420  12    9:6    58                END;
1421  12    9:5    61              IF NOT LSEPPROC THEN
1422  12    9:6    66                WITH LLEXSTK DO
1423  12    9:7    66                  BEGIN SEG := DOLDSEG;
1424  12    9:8    71                    NEXTPROC := SOLDPROC
1425  12    9:7    71                  END;
1426  12    9:5    76              LSEPPROC := FALSE;
1427  12    9:4    79            END;
1428  12    9:2    79        IF NOT MAGIC THEN
1429  12    9:3    83          BEGIN INSYMBOL;
1430  12    9:4    86            TEST := SY <> COMMA;
1431  12    9:4    91            IF TEST THEN
```

```
1432  12    9:5    94              IF SY <> SEMICOLON THEN ERROR(20)
1433  12    9:5    00              ELSE
1434  12    9:4    05            ELSE INSYMBOL
1435  12    9:3    07          END
1436  12    9:1    10      UNTIL TEST OR MAGIC;
1437  12    9:1    15      IF NOT MAGIC THEN
1438  12    9:2    19        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
1439  12    9:1    30      ELSE BEGIN SY := LSY; OP := LOP; ID := LID END;
1440  12    9:1    52      IF NOT USING THEN
1441  12    9:2    57        BEGIN
1442  12    9:3    57          IF INMODULE THEN USINGLIST := NIL;
1443  12    9:3    64          CLOSE(LIBRARY,LOCK);
1444  12    9:3    71          LIBNOTOPEN := TRUE
1445  12    9:2    71        END
1446  12    9:0    74    END (*USESDECLARATION*) ;
1447  12    9:0    94
1448  12   11:D     1    PROCEDURE LABELDECLARATION;
1449  12   11:D     1      VAR LLP: LABELP; REDEF: BOOLEAN;
1450  12   11:0     0    BEGIN
1451  12   11:1     0      REPEAT
1452  12   11:2     0        IF SY = INTCONST THEN
1453  12   11:3     5          WITH DISPLAY[TOP] DO
1454  12   11:4    12            BEGIN LLP := FLABEL; REDEF := FALSE;
1455  12   11:5    19              WHILE (LLP <> NIL) AND NOT REDEF DO
1456  12   11:6    27                IF LLP^.LABVAL <> VAL.IVAL THEN
1457  12   11:7    34                  LLP := LLP^.NEXTLAB
1458  12   11:6    35                ELSE BEGIN REDEF := TRUE; ERROR(166) END;
1459  12   11:5    51              IF NOT REDEF THEN
1460  12   11:6    55                BEGIN NEW(LLP);
1461  12   11:7    60                  WITH LLP^ DO
1462  12   11:8    63                    BEGIN LABVAL := VAL.IVAL;
1463  12   11:9    67                      CODELBP := NIL; NEXTLAB := FLABEL
1464  12   11:8    75                    END;
1465  12   11:7    78                  FLABEL := LLP
1466  12   11:6    81                END;
1467  12   11:5    83              INSYMBOL
1468  12   11:4    83            END
1469  12   11:2    86        ELSE ERROR(15);
1470  12   11:2    92        IF NOT ( SY IN FSYS + [COMMA, SEMICOLON] ) THEN
1471  12   11:3    06          BEGIN ERROR(6); SKIP(FSYS+[COMMA,SEMICOLON]) END;
1472  12   11:2    24        TEST := SY <> COMMA;
```

```
1473  12   11:2   29        IF NOT TEST THEN INSYMBOL
1474  12   11:1   33      UNTIL TEST;
1475  12   11:1   39      IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
1476  12   11:0   50    END (* LABELDECLARATION *) ;
1477  12   11:0   70
1478  12   12:0    1    PROCEDURE CONSTDECLARATION;
1479  12   12:0    1      VAR LCP: CTP; LSP: STP; LVALU: VALU;
1480  12   12:0    0    BEGIN
1481  12   12:1    0      IF SY <> IDENT THEN
1482  12   12:2    5        BEGIN ERROR(2); SKIP(FSYS + [IDENT]) END;
1483  12   12:1   23      WHILE SY = IDENT DO
1484  12   12:2   28        BEGIN NEW(LCP,KONST);
1485  12   12:3   33          WITH LCP^ DO
1486  12   12:4   36            BEGIN NAME := ID; IDTYPE := NIL;
1487  12   12:5   46              NEXT := NIL; KLASS := KONST
1488  12   12:4   54            END;
1489  12   12:3   56          INSYMBOL;
1490  12   12:3   59          IF (SY = RELOP) AND (OP = EQOP) THEN INSYMBOL ELSE ERROR(16);
1491  12   12:3   78          CONSTANT(FSYS + [SEMICOLON],LSP,LVALU);
1492  12   12:3   96          ENTERID(LCP);
1493  12   12:3   00          LCP^.IDTYPE := LSP; LCP^.VALUES := LVALU;
1494  12   12:3   12          IF SY = SEMICOLON THEN
1495  12   12:4   17            BEGIN INSYMBOL;
1496  12   12:5   20              IF NOT (SY IN FSYS + [IDENT]) THEN
1497  12   12:6   34                BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
1498  12   12:4   52            END
1499  12   12:3   52          ELSE
1500  12   12:4   54            IF NOT ((SY = ENDSY) AND (INMODULE)) THEN ERROR(14)
1501  12   12:2   64        END
1502  12   12:0   67    END (*CONSTDECLARATION*) ;
1503  12   12:0   86
1504  12   13:0    1    PROCEDURE TYPEDECLARATION;
1505  12   13:0    1      VAR LCP,LCP1,LCP2: CTP; LSP: STP; LSIZE: ADDRRANGE;
1506  12   13:0    0    BEGIN
1507  12   13:1    0      IF SY <> IDENT THEN
1508  12   13:2    5        BEGIN ERROR(2); SKIP(FSYS + [IDENT]) END;
1509  12   13:1   23      WHILE SY = IDENT DO
1510  12   13:2   28        BEGIN NEW(LCP,TYPES);
1511  12   13:3   33          WITH LCP^ DO
1512  12   13:4   36            BEGIN NAME := ID; IDTYPE := NIL; KLASS := TYPES END;
1513  12   13:3   51          INSYMBOL;
```

```
1514  12   13:3    54          IF (SY = RELOP) AND (OP = EQOP) THEN INSYMBOL ELSE ERROR(16);
1515  12   13:3    73          TYP(FSYS + [SEMICOLON],LSP,LSIZE);
1516  12   13:3    90          ENTERID(LCP);
1517  12   13:3    94          LCP^.IDTYPE := LSP;
1518  12   13:3    99          LCP1 := FWPTR;
1519  12   13:3    03          WHILE LCP1 <> NIL DO
1520  12   13:4    08            BEGIN
1521  12   13:5    08              IF LCP1^.NAME = LCP^.NAME THEN
1522  12   13:6    15                BEGIN
1523  12   13:7    15                  LCP1^.IDTYPE^.ELTYPE := LCP^.IDTYPE;
1524  12   13:7    22                  IF LCP1 <> FWPTR THEN
1525  12   13:8    28                    LCP2^.NEXT := LCP1^.NEXT
1526  12   13:7    32                  ELSE FWPTR := LCP1^.NEXT;
1527  12   13:6    40                END;
1528  12   13:5    40              LCP2 := LCP1; LCP1 := LCP1^.NEXT
1529  12   13:4    44            END;
1530  12   13:3    49          IF SY = SEMICOLON THEN
1531  12   13:4    54            BEGIN INSYMBOL;
1532  12   13:5    57              IF NOT (SY IN FSYS + [IDENT]) THEN
1533  12   13:6    71                BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
1534  12   13:4    89            END
1535  12   13:3    89          ELSE
1536  12   13:4    91            IF NOT ((SY = ENDSY) AND (INMODULE)) THEN ERROR(14)
1537  12   13:2    01        END;
1538  12   13:1    06      IF FWPTR <> NIL THEN
1539  12   13:2    12        BEGIN ERROR(117); FWPTR := NIL END
1540  12   13:0    19    END (*TYPEDECLARATION*) ;
1541  12   13:0    38
1542  12   14:D     1    PROCEDURE VARDECLARATION;
1543  12   14:D     1      VAR LCP,NXT,IDLIST: CTP; LSP: STP; LSIZE: ADDRRANGE;
1544  12   14:0     0    BEGIN NXT := NIL;
1545  12   14:1     3      REPEAT
1546  12   14:2     3        REPEAT
1547  12   14:3     3          IF SY = IDENT THEN
1548  12   14:4     8            BEGIN
1549  12   14:5     8              IF INMODULE THEN NEW(LCP,ACTUALVARS,TRUE)
1550  12   14:5    17              ELSE NEW(LCP,ACTUALVARS,FALSE);
1551  12   14:5    24              WITH LCP^ DO
1552  12   14:6    27                BEGIN NAME := ID; NEXT := NXT; KLASS := ACTUALVARS;
1553  12   14:7    42                  IDTYPE := NIL; VLEV := LEVEL;
1554  12   14:7    53                  IF INMODULE THEN
```

```
1555  12    14:8     57                    IF ININTERFACE THEN PUBLIC := TRUE
1556  12    14:8     64                    ELSE PUBLIC := FALSE
1557  12    14:6     71                  END;
1558  12    14:5     73                ENTERID(LCP);
1559  12    14:5     77                NXT := LCP;
1560  12    14:5     80                INSYMBOL;
1561  12    14:4     83              END
1562  12    14:3     83            ELSE ERROR(2);
1563  12    14:3     89            IF NOT (SY IN FSYS + [COMMA,COLON] + TYPEDELS) THEN
1564  12    14:4     09              BEGIN ERROR(6); SKIP(FSYS+[COMMA,COLON,SEMICOLON]+TYPEDELS) END;
1565  12    14:3     33            TEST := SY <> COMMA;
1566  12    14:3     38            IF NOT TEST THEN INSYMBOL
1567  12    14:2     42          UNTIL TEST;
1568  12    14:2     48          IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1569  12    14:2     62          IDLIST := NXT;
1570  12    14:2     65          TYP(FSYS + [SEMICOLON] + TYPEDELS,LSP,LSIZE);
1571  12    14:2     88          WHILE NXT <> NIL DO
1572  12    14:3     93            WITH  NXT^ DO
1573  12    14:4     96              BEGIN IDTYPE := LSP; VADDR := LC;
1574  12    14:5     06                LC := LC + LSIZE; NXT := NEXT;
1575  12    14:5     15                IF NEXT = NIL THEN
1576  12    14:6     21                  IF LSP <> NIL THEN
1577  12    14:7     26                    IF LSP^.FORM = FILES THEN
1578  12    14:8     32                        BEGIN (*PUT IDLIST INTO LOCAL FILE LIST*)
1579  12    14:9     32                          NEXT := DISPLAY[TOP].FFILE;
1580  12    14:9     42                          DISPLAY[TOP].FFILE := IDLIST
1581  12    14:8     49                        END
1582  12    14:4     51              END;
1583  12    14:2     53          IF SY = SEMICOLON THEN
1584  12    14:3     58            BEGIN INSYMBOL;
1585  12    14:4     61              IF NOT (SY IN FSYS + [IDENT]) THEN
1586  12    14:5     75                BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
1587  12    14:3     93            END
1588  12    14:2     93          ELSE
1589  12    14:3     95              IF NOT ((SY = ENDSY) AND (INMODULE)) THEN ERROR(14)
1590  12    14:1     05      UNTIL (SY <> IDENT) AND NOT (SY IN TYPEDELS);
1591  12    14:1     22    IF FWPTR <> NIL THEN
1592  12    14:2     28        BEGIN ERROR(117); FWPTR := NIL END
1593  12    14:0     35    END (*VARDECLARATION*) ;
1594  12    14:0     54
1595  12    14:0     54
```

```
1596  12   14:0    54 (*$I #5:DECPART.B.TEXT*)
1596  12   14:0    54 (*$I #5:DECPART.C.TEXT*)
1597  12   14:0    54
1598  12   14:0    54 (*    COPYRIGHT (C) 1979, REGENTS OF THE        *)
1599  12   14:0    54 (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
1600  12   14:0    54
1601  12   15:0     1     PROCEDURE PROCDECLARATION(FSY: SYMBOL; SEGDEC: BOOLEAN);
1602  12   15:0     3       VAR LSY: SYMBOL; LCP,LCP1: CTP; LSP: STP;
1603  12   15:0     7           EXTONLY,FORW: BOOLEAN;
1604  12   15:0     9           LCM: ADDRRANGE;
1605  12   15:0    10           LLEXSTK: LEXSTKREC;
1606  12   15:0    21
1607  12   16:0     1       PROCEDURE PARAMETERLIST(FSY: SETOFSYS; VAR FPAR: CTP; FCP: CTP);
1608  12   16:0     7         VAR LCP,LCP1,LCP2,LCP3: CTP; LSP: STP; LKIND: IDKIND;
1609  12   16:0    13           LLC,LEN : ADDRRANGE; COUNT : INTEGER;
1610  12   16:0     0       BEGIN LCP1 := NIL; LLC := LC;
1611  12   16:1     6         IF NOT (SY IN FSY + [LPARENT]) THEN
1612  12   16:2    19           BEGIN ERROR(7); SKIP(FSYS + FSY + [LPARENT]) END;
1613  12   16:1    43         IF SY = LPARENT THEN
1614  12   16:2    48           BEGIN IF FORW THEN ERROR(119);
1615  12   16:3    57             INSYMBOL;
1616  12   16:3    60             IF NOT (SY IN [IDENT,VARSY]) THEN
1617  12   16:4    73               BEGIN ERROR(7); SKIP(FSYS + [IDENT,RPARENT]) END;
1618  12   16:3    91             WHILE SY IN [IDENT,VARSY] DO
1619  12   16:4    02               BEGIN
1620  12   16:5    02                 IF SY = VARSY THEN
1621  12   16:6    07                   BEGIN LKIND := FORMAL; INSYMBOL END
1622  12   16:5    13                 ELSE LKIND := ACTUAL;
1623  12   16:5    18                 LCP2 := NIL;
1624  12   16:5    21                 COUNT := 0;
1625  12   16:5    24                 REPEAT
1626  12   16:6    24                   IF SY <> IDENT THEN ERROR(2)
1627  12   16:6    30                   ELSE
1628  12   16:7    35                     BEGIN
1629  12   16:8    35                       NEW(LCP,FORMALVARS,FALSE); (*MAY BE ACTUAL(SAME SIZE)*)
1630  12   16:8    40                       WITH LCP^ DO
1631  12   16:9    43                         BEGIN NAME := ID; IDTYPE := NIL; NEXT := LCP2;
1632  12   16:0    58                           IF LKIND = FORMAL THEN KLASS := FORMALVARS
1633  12   16:0    66                           ELSE KLASS := ACTUALVARS; VLEV := LEVEL
1634  12   16:9    78                         END;
1635  12   16:8    81                       ENTERID(LCP);
```

```
1636  12    16:8     85          LCP2 := LCP; COUNT := COUNT + 1;
1637  12    16:8     93          INSYMBOL
1638  12    16:7     93        END;
1639  12    16:6     96      IF NOT (SY IN FSYS + [COMMA,SEMICOLON,COLON]) THEN
1640  12    16:7     10        BEGIN ERROR(7);
1641  12    16:8     14          SKIP(FSYS + [COMMA,SEMICOLON,RPARENT,COLON])
1642  12    16:7     25        END;
1643  12    16:6     28      TEST := SY <> COMMA;
1644  12    16:6     33      IF NOT TEST THEN INSYMBOL
1645  12    16:5     37    UNTIL TEST;
1646  12    16:5     43    LSP := NIL;
1647  12    16:5     46    IF SY = COLON THEN
1648  12    16:6     51      BEGIN INSYMBOL;
1649  12    16:7     54        IF SY = IDENT THEN
1650  12    16:8     59          BEGIN
1651  12    16:9     59            SEARCHID([TYPES],LCP);
1652  12    16:9     68            INSYMBOL;
1653  12    16:9     71            LSP := LCP^.IDTYPE;
1654  12    16:9     75            LEN := PTRSIZE;
1655  12    16:9     78            IF LSP <> NIL THEN
1656  12    16:0     83              IF LKIND = ACTUAL THEN
1657  12    16:1     88                IF LSP^.FORM = FILES THEN ERROR(121)
1658  12    16:1     95                ELSE
1659  12    16:2     00                  IF LSP^.FORM <= POWER THEN LEN := LSP^.SIZE;
1660  12    16:9     10            LC := LC + COUNT * LEN
1661  12    16:8     12          END
1662  12    16:7     17        ELSE ERROR(2)
1663  12    16:6     20      END
1664  12    16:5     23    ELSE
1665  12    16:6     25      IF LKIND = FORMAL THEN
1666  12    16:7     30        EXTONLY := TRUE
1667  12    16:6     30      ELSE ERROR(5);
1668  12    16:5     40    IF NOT (SY IN FSYS + [SEMICOLON,RPARENT]) THEN
1669  12    16:6     54      BEGIN ERROR(7); SKIP(FSYS + [SEMICOLON,RPARENT]) END;
1670  12    16:5     72    LCP3 := LCP2; LCP := NIL;
1671  12    16:5     78    WHILE LCP2 <> NIL DO
1672  12    16:6     83      BEGIN LCP := LCP2;
1673  12    16:7     86        WITH LCP2^ DO
1674  12    16:8     89          BEGIN IDTYPE := LSP;
1675  12    16:9     94            LCP2 := NEXT
1676  12    16:8     94          END
```

```
1677  12   16:6    98                    END;
1678  12   16:5    00                  IF LCP <> NIL THEN
1679  12   16:6    05                    BEGIN LCP^.NEXT := LCP1; LCP1 := LCP3 END;
1680  12   16:5    13                  IF SY = SEMICOLON THEN
1681  12   16:6    18                    BEGIN INSYMBOL;
1682  12   16:7    21                      IF NOT (SY IN FSYS + [IDENT,VARSY]) THEN
1683  12   16:8    40                        BEGIN ERROR(7); SKIP(FSYS + [IDENT,RPARENT]) END
1684  12   16:6    58                    END
1685  12   16:4    58                END (*WHILE*) ;
1686  12   16:3    60              IF SY = RPARENT THEN
1687  12   16:4    65                BEGIN INSYMBOL;
1688  12   16:5    68                  IF NOT (SY IN FSY + FSYS) THEN
1689  12   16:6    85                    BEGIN ERROR(6); SKIP(FSY + FSYS) END
1690  12   16:4    06                END
1691  12   16:3    06              ELSE ERROR(4);
1692  12   16:3    12              FCP^.LOCALLC := LC; LCP3 := NIL;
1693  12   16:3    20              WHILE LCP1 <> NIL DO
1694  12   16:4    25                WITH LCP1^ DO
1695  12   16:5    28                  BEGIN LCP2 := NEXT; NEXT := LCP3;
1696  12   16:6    37                    IF (IDTYPE <> NIL) THEN
1697  12   16:7    43                      IF KLASS = FORMALVARS THEN
1698  12   16:8    50                        BEGIN VADDR := LLC; LLC := LLC + PTRSIZE END
1699  12   16:7    60                      ELSE
1700  12   16:8    62                        IF KLASS = ACTUALVARS THEN
1701  12   16:9    69                          IF (IDTYPE^.FORM <= POWER) THEN
1702  12   16:0    76                            BEGIN VADDR := LLC; LLC := LLC + IDTYPE^.SIZE END
1703  12   16:9    88                          ELSE
1704  12   16:0    90                            BEGIN VADDR := LC;
1705  12   16:1    95                              LC := LC + IDTYPE^.SIZE;
1706  12   16:1    02                              LLC := LLC + PTRSIZE
1707  12   16:0    03                            END;
1708  12   16:6    07                    LCP3 := LCP1; LCP1 := LCP2
1709  12   16:5    10                  END;
1710  12   16:3    15              FPAR := LCP3
1711  12   16:2    16            END
1712  12   16:1    18              ELSE FPAR := NIL
1713  12   16:0    21      END (*PARAMETERLIST*) ;
1714  12   16:0    48
1715  12   15:0     0      BEGIN (*PROCDECLARATION*)
1716  12   15:1     0        IF SEGDEC THEN (* SEGMENT DECLARATION *)
1717  12   15:2     3          BEGIN
```

```
1718  12   15:3    3          IF CODEINSEG THEN
1719  12   15:4    7            BEGIN ERROR(399); SEGINX:=0; CURBYTE:=0; END;
1720  12   15:3   20          WITH LLEXSTK DO
1721  12   15:4   20            BEGIN
1722  12   15:5   20              OLDSEG:=SEG;
1723  12   15:5   23              SEG:=NEXTSEG;
1724  12   15:5   27              OLDPROC:=NEXTPROC;
1725  12   15:4   31            END;
1726  12   15:3   31          NEXTPROC:=1;
1727  12   15:3   34          LSY:=SY;
1728  12   15:3   37          IF SY IN [PROCSY,FUNCSY] THEN INSYMBOL
1729  12   15:3   50          ELSE BEGIN ERROR(399); LSY:=PROCSY END;
1730  12   15:3   64          FSY:=LSY;
1731  12   15:2   67        END;
1732  12   15:1   67      LLEXSTK.DLLC := LC; LC := LCAFTERMARKSTACK;
1733  12   15:1   73      IF FSY = FUNCSY THEN LC := LC + REALSIZE;
1734  12   15:1   83      LINEINFO := LC; DP := TRUE; EXTONLY := FALSE;
1735  12   15:1   92      IF SY = IDENT THEN
1736  12   15:2   97        BEGIN
1737  12   15:3   97          IF USING OR INMODULE AND ININTERFACE THEN FORW := FALSE
1738  12   15:3   07          ELSE
1739  12   15:4   12            BEGIN SEARCHSECTION(DISPLAY[TOP].FNAME,LCP);
1740  12   15:5   23              IF LCP <> NIL THEN
1741  12   15:6   28                BEGIN
1742  12   15:7   28                  IF LCP^.KLASS = PROC THEN
1743  12   15:8   35                    FORW := LCP^.FORWDECL AND (FSY = PROCSY)
1744  12   15:8   41                            AND (LCP^.PFKIND = ACTUAL)
1745  12   15:7   47                  ELSE
1746  12   15:8   52                    IF LCP^.KLASS = FUNC THEN
1747  12   15:9   59                      FORW := LCP^.FORWDECL AND (FSY = FUNCSY)
1748  12   15:9   65                              AND (LCP^.PFKIND = ACTUAL)
1749  12   15:8   71                    ELSE FORW := FALSE;
1750  12   15:7   79                  IF NOT FORW THEN ERROR(160)
1751  12   15:6   86                END
1752  12   15:5   89              ELSE FORW := FALSE
1753  12   15:4   91            END;
1754  12   15:3   94          IF NOT FORW THEN
1755  12   15:4   98            BEGIN
1756  12   15:5   98              IF FSY = PROCSY THEN
1757  12   15:6   03                IF INMODULE THEN NEW(LCP,PROC,DECLARED,ACTUAL,TRUE)
1758  12   15:6   12                ELSE NEW(LCP,PROC,DECLARED,ACTUAL,FALSE)
```

```
1759  12   15:5   19              ELSE
1760  12   15:6   21                IF INMODULE THEN NEW(LCP,FUNC,DECLARED,ACTUAL,TRUE)
1761  12   15:6   30                ELSE NEW(LCP,FUNC,DECLARED,ACTUAL,FALSE);
1762  12   15:5   37              WITH LCP^ DO
1763  12   15:6   40                BEGIN NAME := ID; IDTYPE := NIL; LOCALLC := LC;
1764  12   15:7   58                  PFDECKIND := DECLARED; PFKIND := ACTUAL;
1765  12   15:7   70                  INSCOPE := FALSE; PFLEV := LEVEL;
1766  12   15:7   83                  PFNAME := NEXTPROC; PFSEG := SEG;
1767  12   15:7   96                  IF USING THEN PROCTABLE[NEXTPROC] := 0;
1768  12   15:7   09                  IF INMODULE THEN
1769  12   15:8   13                    IF USING THEN IMPORTED := TRUE
1770  12   15:8   21                    ELSE IMPORTED := FALSE;
1771  12   15:7   31                  IF SEGDEC THEN
1772  12   15:8   34                    BEGIN
1773  12   15:9   34                      IF NEXTSEG > MAXSEG THEN ERROR(250);
1774  12   15:9   46                      NEXTSEG := NEXTSEG+1;
1775  12   15:9   52                      SEGTABLE[SEG].SEGNAME := ID
1776  12   15:8   60                    END;
1777  12   15:7   64                  IF NEXTPROC = MAXPROCNUM THEN ERROR(251)
1778  12   15:7   75                  ELSE NEXTPROC := NEXTPROC + 1;
1779  12   15:7   86                  IF FSY = PROCSY THEN KLASS := PROC
1780  12   15:7   95                  ELSE KLASS := FUNC
1781  12   15:6   03                END;
1782  12   15:5   05              ENTERID(LCP)
1783  12   15:4   06            END
1784  12   15:3   09          ELSE
1785  12   15:4   11            BEGIN LCP1 := LCP^.NEXT;
1786  12   15:5   15              WHILE LCP1 <> NIL DO
1787  12   15:6   20                BEGIN
1788  12   15:7   20                  WITH LCP1^ DO
1789  12   15:8   23                    IF IDTYPE = NIL THEN
1790  12   15:9   30                      EXTONLY := TRUE
1791  12   15:8   30                    ELSE
1792  12   15:9   35                      IF KLASS = FORMALVARS THEN
1793  12   15:0   43                        BEGIN
1794  12   15:1   43                          LCM := VADDR + PTRSIZE;
1795  12   15:1   51                          IF LCM > LC THEN LC := LCM
1796  12   15:0   56                        END
1797  12   15:9   59                      ELSE
1798  12   15:0   61                        IF KLASS = ACTUALVARS THEN
1799  12   15:1   69                          BEGIN
```

```
1800  12   15:2   69                              LCM := VADDR + IDTYPE^.SIZE;
1801  12   15:2   80                              IF LCM > LC THEN LC := LCM
1802  12   15:1   85                            END;
1803  12   15:7   88                        LCP1 := LCP1^.NEXT
1804  12   15:6   89                      END;
1805  12   15:5   94                    IF SEG <> LCP^.PFSEG THEN
1806  12   15:6   01                      BEGIN
1807  12   15:7   01                        SEG := LCP^.PFSEG; NEXTPROC := 2;
1808  12   15:7   09                        IF NOT SEGDEC THEN ERROR(399)
1809  12   15:6   16                      END
1810  12   15:4   19                END;
1811  12   15:3   19              INSYMBOL
1812  12   15:2   19            END
1813  12   15:1   22          ELSE
1814  12   15:2   24            BEGIN ERROR(2); LCP := UPRCPTR END;
1815  12   15:1   32          WITH LLEXSTK DO
1816  12   15:2   32            BEGIN DOLDLEV:=LEVEL;
1817  12   15:3   36              DOLDTOP:=TOP;
1818  12   15:3   39              POLDPROC:=CURPROC;
1819  12   15:3   43              DFPROCP:=LCP;
1820  12   15:2   46            END;
1821  12   15:1   46          CURPROC := LCP^.PFNAME;
1822  12   15:1   51          IF LEVEL < MAXLEVEL THEN LEVEL := LEVEL + 1 ELSE ERROR(251);
1823  12   15:1   71          IF TOP < DISPLIMIT THEN
1824  12   15:2   76            BEGIN TOP := TOP + 1;
1825  12   15:3   81              WITH DISPLAY[TOP] DO
1826  12   15:4   88                BEGIN
1827  12   15:5   88                  IF FORW THEN FNAME := LCP^.NEXT
1828  12   15:5   94                  ELSE FNAME := NIL;
1829  12   15:5   02                  FLABEL := NIL; FFILE := NIL; OCCUR := BLCK
1830  12   15:4   18                END
1831  12   15:2   20            END
1832  12   15:1   20          ELSE ERROR(250);
1833  12   15:1   28          IF FSY = PROCSY THEN
1834  12   15:2   33            BEGIN PARAMETERLIST([SEMICOLON],LCP1,LCP);
1835  12   15:3   42              IF NOT FORW THEN LCP^.NEXT := LCP1
1836  12   15:2   49            END
1837  12   15:1   51          ELSE
1838  12   15:2   53            BEGIN PARAMETERLIST([SEMICOLON,COLON],LCP1,LCP);
1839  12   15:3   62              IF NOT FORW THEN LCP^.NEXT := LCP1;
1840  12   15:3   71              IF SY = COLON THEN
```

```
1841  12   15:4   76              BEGIN INSYMBOL;
1842  12   15:5   79                IF SY = IDENT THEN
1843  12   15:6   84                  BEGIN IF FORW THEN ERROR(122);
1844  12   15:7   91                    SEARCHID([TYPES],LCP1);
1845  12   15:7   00                    LSP := LCP1^.IDTYPE;
1846  12   15:7   04                    LCP^.IDTYPE := LSP;
1847  12   15:7   09                    IF LSP <> NIL THEN
1848  12   15:8   14                      IF NOT (LSP^.FORM IN [SCALAR,SUBRANGE,POINTER]) THEN
1849  12   15:9   22                        BEGIN ERROR(120); LCP^.IDTYPE := NIL END;
1850  12   15:7   31                    INSYMBOL
1851  12   15:6   31                  END
1852  12   15:5   34                ELSE BEGIN ERROR(2); SKIP(FSYS + [SEMICOLON]) END
1853  12   15:4   54              END
1854  12   15:3   54            ELSE
1855  12   15:4   56              IF NOT FORW THEN ERROR(123)
1856  12   15:2   61          END;
1857  12   15:1   64        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
1858  12   15:1   78        LCP^.EXTURNAL := FALSE;
1859  12   15:1   83        IF (SY = EXTERNLSY)
1860  12   15:1   86           OR ((USING) AND (LSEPPROC)) THEN
1861  12   15:2   94          BEGIN
1862  12   15:3   94            IF LEVEL <> 2 THEN
1863  12   15:4   00              ERROR(183) (*EXTERNAL PROCS MUST BE IN OUTERMOST BLOCK*);
1864  12   15:3   06            IF INMODULE THEN
1865  12   15:4   10              IF ININTERFACE AND NOT USING THEN
1866  12   15:5   18                ERROR(184); (*NO EXTERNAL DECL IN INTERFACE*)
1867  12   15:3   24            IF SEGDEC THEN ERROR(399);
1868  12   15:3   33            WITH LCP^ DO
1869  12   15:4   36              BEGIN EXTURNAL := TRUE; FORWDECL := FALSE;
1870  12   15:5   48                WRITELN(OUTPUT); WRITELN(OUTPUT,NAME,' [',MEMAVAIL:5,' WORDS]');
1871  12   15:5   08                WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
1872  12   15:4   33              END;
1873  12   15:3   33            PROCTABLE[CURPROC] := 0;
1874  12   15:3   42            DLINKERINFO := TRUE;
1875  12   15:3   45            IF SY = EXTERNLSY THEN
1876  12   15:4   50              BEGIN INSYMBOL;
1877  12   15:5   53                IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
1878  12   15:5   67                IF NOT (SY IN FSYS) THEN
1879  12   15:6   78                  BEGIN ERROR(6); SKIP(FSYS) END
1880  12   15:4   93              END
1881  12   15:2   93          END
```

```
1882  12   15:1     93        ELSE
1883  12   15:2     95          IF USING THEN
1884  12   15:3     99            BEGIN LCP^.FORWDECL := FALSE;
1885  12   15:3   1 04            END
1886  12   15:2   1 04          ELSE
1887  12   15:3   1 06            IF (SY = FORWARDSY) OR INMODULE AND ININTERFACE THEN
1888  12   15:4   1 17              BEGIN
1889  12   15:5   1 17                IF FORW THEN ERROR(161)
1890  12   15:5   1 23                ELSE LCP^.FORWDECL := TRUE;
1891  12   15:5   1 33                IF SY = FORWARDSY THEN
1892  12   15:6   1 38                  BEGIN INSYMBOL;
1893  12   15:7   1 41                    IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
1894  12   15:6   1 52                  END;
1895  12   15:5   1 55                IF NOT (SY IN FSYS) THEN
1896  12   15:6   1 66                  BEGIN ERROR(6); SKIP(FSYS) END
1897  12   15:4   1 81              END
1898  12   15:3   1 81            ELSE
1899  12   15:4   1 83              BEGIN
1900  12   15:5   1 83                IF EXTONLY THEN
1901  12   15:6   1 86                  ERROR(7);
1902  12   15:5   1 90                NEWBLOCK:=TRUE;
1903  12   15:5   1 93                NOTDONE:=TRUE;
1904  12   15:5   1 97                WITH LLEXSTK DO
1905  12   15:6   1 97                  BEGIN
1906  12   15:7   1 97                    MARK(DMARKP);
1907  12   15:7   1 01                    WITH LCP^ DO
1908  12   15:8   1 04                      BEGIN FORWDECL := FALSE; INSCOPE := TRUE;
1909  12   15:9   1 16                        EXTURNAL := FALSE END;
1910  12   15:7   1 22                    BFSY:=SEMICOLON;
1911  12   15:7   1 25                    ISSEGMENT:=SEGDEC;
1912  12   15:7   1 28                    PREVLEXSTACKP:=TOS;
1913  12   15:6   1 32                   END;
1914  12   15:5   1 32                NEW(TOS);
1915  12   15:5   1 37                TOS^:=LLEXSTK;
1916  12   15:5   1 43                EXIT(PROCDECLARATION);
1917  12   15:4   1 47              END;
1918  12   15:1   1 47      WITH LLEXSTK DO  (* FORWARD OR EXTERNAL DECLARATION, SO RESTORE STATE *)
1919  12   15:2   1 47        BEGIN
1920  12   15:3   1 47          LEVEL:=DOLDLEV;
1921  12   15:3   1 50          TOP:=DOLDTOP;
1922  12   15:3   1 53          LC:=DLLC;
```

```
1923  12    15:3   1 56            CURPROC:=POLDPROC;
1924  12    15:3   1 59            IF SEGDEC THEN
1925  12    15:4   1 62              BEGIN
1926  12    15:5   1 62                NEXTPROC:=SOLDPROC;
1927  12    15:5   1 65                SEG:=DOLDSEG;
1928  12    15:4   1 68              END;
1929  12    15:2   1 68          END;
1930  12    15:0   1 68       END; (* PROCDECLARATION *)
1931  12    15:0   1 92
1932  12    15:0   1 92
1933  12     1:0     0    BEGIN (*DECLARATIONPART*)
1934  12     1:1     0      IF (NOSWAP) AND (STARTINGUP) THEN
1935  12     1:2     7        BEGIN
1936  12     1:3     7          STARTINGUP:=FALSE; (* ALL SEGMENTS ARE IN BY THIS TIME *)
1937  12     1:3    10          BLOCK(FSYS);
1938  12     1:3    20          EXIT(DECLARATIONPART);
1939  12     1:2    24        END;
1940  12     1:1    24      IF NOISY THEN
1941  12     1:2    28        UNITWRITE(3,DUMMYVAR[-1600],35); (*ADJUST DISPLAY OF STACK AND HEAP*)
1942  12     1:1    43      REPEAT
1943  12     1:2    43        NOTDONE:=FALSE;
1944  12     1:2    46        IF USERINFO.STUPID THEN
1945  12     1:3    50          IF NOT CODEINSEG THEN
1946  12     1:4    55            IF (LEVEL = 1) AND (NEXTSEG = 10) THEN
1947  12     1:5    66              IF NOT(INMODULE OR USING) THEN USESDECLARATION(TRUE);
1948  12     1:5    77              (*TO GET TURTLE GRAPHICS*)
1949  12     1:2    77        IF SY = USESSY THEN
1950  12     1:3    82          BEGIN INSYMBOL; USESDECLARATION(FALSE) END;
1951  12     1:2    88        IF SY = LABELSY THEN
1952  12     1:3    93          BEGIN
1953  12     1:4    93            IF INMODULE AND ININTERFACE THEN
1954  12     1:5    00              BEGIN ERROR(186); SKIP(FSYS - [LABELSY]) END
1955  12     1:4    25            ELSE INSYMBOL; LABELDECLARATION END;
1956  12     1:2    32        IF SY = CONSTSY THEN
1957  12     1:3    37          BEGIN INSYMBOL; CONSTDECLARATION END;
1958  12     1:2    42        IF SY = TYPESY THEN
1959  12     1:3    47          BEGIN INSYMBOL; TYPEDECLARATION END;
1960  12     1:2    52        IF SY = VARSY THEN
1961  12     1:3    57          BEGIN INSYMBOL; VARDECLARATION END;
1962  12     1:2    62        IF LEVEL = 1 THEN GLEV := TOP;
1963  12     1:2    71        IF SY IN [PROCSY,FUNCSY,PROGSY] THEN
```

```
1964  12    1:3    84          BEGIN
1965  12    1:4    84            IF INMODULE THEN
1966  12    1:5    88              IF ININTERFACE AND NOT USING THEN PUBLICPROCS := TRUE;
1967  12    1:4    99            REPEAT
1968  12    1:5    99              LSY := SY; INSYMBOL;
1969  12    1:5    05              IF LSY = PROGSY THEN
1970  12    1:6    10                IF INMODULE THEN
1971  12    1:7    14                  BEGIN ERROR(185 (*SEG DEC NOT ALLOWED IN UNIT*));
1972  12    1:8    20                    PROCDECLARATION(PROCSY,FALSE)
1973  12    1:7    22                  END
1974  12    1:6    24                ELSE PROCDECLARATION(LSY,TRUE)
1975  12    1:5    28              ELSE PROCDECLARATION(LSY,FALSE);
1976  12    1:4    36            UNTIL NOT (SY IN [PROCSY,FUNCSY,PROGSY])
1977  12    1:3    48          END;
1978  12    1:2    51        IF (SY <> BEGINSY) THEN
1979  12    1:3    56          IF NOT ((USING OR INMODULE) AND (SY IN [IMPLESY,ENDSY]))
1980  12    1:3    75             AND NOT( SY IN [SEPARATSY,UNITSY]) THEN
1981  12    1:4    94            IF (NOT (INCLUDING OR NOTDONE))
1982  12    1:4    99              OR
1983  12    1:4    99              NOT(SY IN BLOCKBEGSYS) THEN
1984  12    1:5    10             BEGIN ERROR(18); SKIP(FSYS - [UNITSY,INTERSY]); END;
1985  12    1:1    37      UNTIL (SY IN (STATBEGSYS + [SEPARATSY,UNITSY,IMPLESY,ENDSY]));
1986  12    1:1    59      NEWBLOCK:=FALSE;
1987  12    1:0    62    END (*DECLARATIONPART*) ;
1988  12    1:0    78
1989  12    1:0    78
1990  12    1:0    78  (*$I #5:DECPART.C.TEXT*)
1990  12    1:0    78  (*$I #5:BODYPART.A.TEXT*)
1991  12    1:0    78
1992  12    1:0    78  (*    COPYRIGHT (C) 1979, REGENTS OF THE          *)
1993  12    1:0    78  (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO         *)
1994  12    1:0    78
1995  13    1:D     1  SEGMENT PROCEDURE BODYPART(FSYS: SETOFSYS; FPROCP: CTP);
1996  13    1:D     6
1997  13    2:D     1    PROCEDURE LINKERREF(KLASS: IDCLASS; ID,ADDR: INTEGER);
1998  13    2:0     0    BEGIN
1999  13    2:1     0      IF NREFS > REFSPERBLK THEN (*WRITE BUFFER*)
2000  13    2:2     9        BEGIN
2001  13    2:3     9          IF BLOCKWRITE(REFFILE,REFLIST^,1,REFBLK) <> 1 THEN ERROR(402);
2002  13    2:3    36          REFBLK := REFBLK + 1;
2003  13    2:3    44          NREFS := 1
```

```
2004  13    2:2    44        END;
2005  13    2:1    48      WITH REFLIST^[NREFS] DO
2006  13    2:2    60        BEGIN
2007  13    2:3    60          IF KLASS IN VARS THEN KEY := ID + 32
2008  13    2:3    71          ELSE (*PROC*) KEY := ID;
2009  13    2:3    81          OFFSET := SEGINX + ADDR
2010  13    2:2    84        END;
2011  13    2:1    87      NREFS := NREFS + 1
2012  13    2:0    90    END (*LINKERREF*) ;
2013  13    2:0    08
2014  13    3:D     1    PROCEDURE GEN0(FOP: OPRANGE);
2015  13    3:D     2      VAR I: INTEGER; ODDIC: BOOLEAN;
2016  13    3:0     0    BEGIN
2017  13    3:1     0      IF FOP <> 38(*LCA*) THEN GENBYTE(FOP+128)
2018  13    3:1    10      ELSE
2019  13    3:2    15        BEGIN
2020  13    3:3    15          ODDIC := ODD(IC); STRGCSTIC := IC;
2021  13    3:3    21          IF NOT ODDIC THEN GENBYTE(215(*NOP*));
2022  13    3:3    31          GENBYTE(166(*LCA*));
2023  13    3:3    37          WITH GATTR.CVAL.VALP^ DO
2024  13    3:4    40            BEGIN GENBYTE(SLGTH);
2025  13    3:5    45              FOR I := 1 TO SLGTH DO GENBYTE(ORD(SVAL[I]));
2026  13    3:5    74              IF ODDIC THEN GENBYTE(215(*NOP*))
2027  13    3:4    80            END
2028  13    3:2    83        END
2029  13    3:0    83    END (*GEN0*) ;
2030  13    3:0    98
2031  13    4:D     1    PROCEDURE GENLDC(IVAL: INTEGER);
2032  13    4:0     0    BEGIN
2033  13    4:1     0      IF (IVAL >= 0) AND (IVAL <= 127) THEN GENBYTE(IVAL)
2034  13    4:1    10      ELSE
2035  13    4:2    15        BEGIN GENBYTE(51(*LDC*)+148);
2036  13    4:3    23          GENBYTE( ABS(IVAL) MOD 256 );
2037  13    4:3    32          GENBYTE( ABS(IVAL) DIV 256 );
2038  13    4:3    41          IF IVAL<0 THEN GEN0(17(*NGI*))
2039  13    4:2    47        END
2040  13    4:0    49    END (*GENLDC*) ;
2041  13    4:0    62
2042  13    5:D     1    PROCEDURE GENBIG(IVAL: INTEGER);
2043  13    5:0     0    BEGIN
2044  13    5:1     0      IF IVAL <= 127 THEN GENBYTE(IVAL)
```

```
2045  13    5:1     6      ELSE
2046  13    5:2    11        BEGIN
2047  13    5:3    11          GENBYTE( 128+(IVAL DIV 256) );
2048  13    5:3    23          GENBYTE( IVAL MOD 256 )
2049  13    5:2    28        END
2050  13    5:0    31    END (*GENBIG*) ;
2051  13    5:0    44
2052  13    6:D     1    PROCEDURE GEN1(FOP: OPRANGE; FP2: INTEGER);
2053  13    6:D     3      LABEL 1;
2054  13    6:D     3      VAR I,J: INTEGER;
2055  13    6:0     0    BEGIN
2056  13    6:1     0      GENBYTE(FOP+128);
2057  13    6:1     8      IF FOP = 51(*LDC*) THEN
2058  13    6:2    13        BEGIN
2059  13    6:3    13          IF FP2 = 2 THEN I := REALSIZE
2060  13    6:3    18          ELSE
2061  13    6:4    23            BEGIN I := 8;
2062  13    6:5    26              WHILE I > 0 DO
2063  13    6:6    31                IF GATTR.CVAL.VALP^.CSTVAL[I] <> 0 THEN GOTO 1
2064  13    6:6    46                ELSE I := I - 1;
2065  13    6:5    55        1:    END;
2066  13    6:3    55          GATTR.TYPTR^.SIZE := I;
2067  13    6:3    58          IF I > 1 THEN
2068  13    6:4    63            BEGIN GENBYTE(I);
2069  13    6:5    67              FOR J := I DOWNTO 1 DO GENWORD(GATTR.CVAL.VALP^.CSTVAL[J])
2070  13    6:4    87            END
2071  13    6:3    97          ELSE
2072  13    6:4    99            BEGIN IC := IC - 1;
2073  13    6:5    04              IF I = 1 THEN GENLDC(GATTR.CVAL.VALP^.CSTVAL[1])
2074  13    6:4    18            END
2075  13    6:2    20        END
2076  13    6:1    20      ELSE
2077  13    6:2    22        IF FOP IN [30(*CSP*),32(*ADJ*),45(*RNP*),
2078  13    6:2    23                   46(*CIP*),60(*LDM*),61(*STM*),
2079  13    6:2    23                   65(*RBP*),66(*CBP*),78(*CLP*),
2080  13    6:2    23                   42(*SAS*),79(*CGP*)] THEN GENBYTE(FP2)
2081  13    6:2    41        ELSE
2082  13    6:3    46          IF INMODULE AND (FOP IN [37(*LAO*),41(*LDO*),43(*SRO*)]) THEN
2083  13    6:4    63            BEGIN LINKERREF(ACTUALVARS,FP2,IC); GENBYTE(128); GENBYTE(0) END
2084  13    6:3    78          ELSE
2085  13    6:4    80            IF ((FOP = 74(*LDL*)) OR (FOP = 41(*LDO*)))
```

```
2086  13    6:4    87              AND (FP2 <= 16) THEN
2087  13    6:5    93            BEGIN IC := IC-1;
2088  13    6:6    98              IF FOP = 41(*LDO*) THEN GENBYTE(231+FP2)
2089  13    6:6    08              ELSE GENBYTE(215+FP2)
2090  13    6:5    18            END
2091  13    6:4    21          ELSE
2092  13    6:5    23            IF (FOP = 35(*IND*)) AND (FP2 <= 7) THEN
2093  13    6:6    32              BEGIN IC := IC-1; GENBYTE(248+FP2) END
2094  13    6:5    45            ELSE
2095  13    6:6    47              GENBIG(FP2)
2096  13    6:0    48  END (*GEN1*) ;
2097  13    6:0    68
2098  13    7:D     1  PROCEDURE GEN2(FOP: OPRANGE; FP1,FP2: INTEGER);
2099  13    7:0     0  BEGIN
2100  13    7:1     0    IF (FOP = 64(*IXP*)) OR (FOP = 77(*CXP*)) THEN
2101  13    7:2     9      BEGIN GENBYTE(FOP+128); GENBYTE(FP1); GENBYTE(FP2);
2102  13    7:2    25      END
2103  13    7:1    25    ELSE
2104  13    7:2    27      IF FOP IN [47(*EQU*),48(*GEQ*),49(*GRT*),
2105  13    7:2    28                 52(*LEQ*),53(*LES*),55(*NEQ*)] THEN
2106  13    7:3    42        IF FP1 = 0 THEN GEN0(FOP+20)
2107  13    7:3    50        ELSE
2108  13    7:4    54          BEGIN GEN1(FOP,FP1+FP1);
2109  13    7:5    60            IF FP1 > 4 THEN GENBIG(FP2)
2110  13    7:4    66          END
2111  13    7:2    68      ELSE
2112  13    7:3    70        BEGIN (*LDA,LOD,STR*)
2113  13    7:4    70          IF FP1 = 0 THEN GEN1(FOP+20,FP2)
2114  13    7:4    79          ELSE
2115  13    7:5    83            BEGIN
2116  13    7:6    83              GENBYTE(FOP+128); GENBYTE(FP1); GENBIG(FP2)
2117  13    7:5    96            END
2118  13    7:3    98        END;
2119  13    7:0    98  END (*GEN2*) ;
2120  13    7:0    10
2121  13    8:D     1  PROCEDURE GENNR(EXTPROC: NONRESIDENT);
2122  13    8:D     2
2123  13    9:D     1   PROCEDURE ASSIGN(EXTPROC: NONRESIDENT);
2124  13    9:0     0   BEGIN
2125  13    9:1     0     PROCTABLE[NEXTPROC] := 0;
2126  13    9:1     9     PFNUMOF[EXTPROC] := NEXTPROC; NEXTPROC := NEXTPROC + 1;
```

```
2127  13    9:1   24      IF NEXTPROC > MAXPROCNUM THEN ERROR(193);(*NOT ENOUGH ROOM FOR THIS*)
2128  13    9:1   38      CLINKERINFO := TRUE
2129  13    9:0   38    END (*ASSIGN*) ;                                          (*OPERATION*)
2130  13    9:0   54
2131  13    8:0    0  BEGIN (*GENNR*)
2132  13    8:1    0    IF PFNUMOF[EXTPROC] = 0 THEN ASSIGN(EXTPROC);
2133  13    8:1   14    IF SEPPROC THEN
2134  13    8:2   18      BEGIN
2135  13    8:3   18        GEN1(79(*CGP*),0); LINKERREF(PROC,-PFNUMOF[EXTPROC],IC-1)
2136  13    8:2   34      END
2137  13    8:1   36    ELSE
2138  13    8:2   38      GEN1(79(*CGP*),PFNUMOF[EXTPROC]);
2139  13    8:0   48  END (*GENNR*) ;
2140  13    8:0   60
2141  13   10:0    1  PROCEDURE GENJMP(FOP: OPRANGE; FLBP: LBP);
2142  13   10:0    3    VAR DISP: INTEGER;
2143  13   10:0    0  BEGIN
2144  13   10:1    0    WITH FLBP^ DO
2145  13   10:2    3      IF DEFINED THEN
2146  13   10:3    7        BEGIN
2147  13   10:4    7          GENBYTE(FOP+128);
2148  13   10:4   15          DISP := OCCURIC-IC-1;
2149  13   10:4   23          IF (DISP >= 0) AND (DISP <= 127) THEN GENBYTE(DISP)
2150  13   10:4   33          ELSE
2151  13   10:5   38            BEGIN
2152  13   10:6   38              IF JTABINX = 0 THEN
2153  13   10:7   44                BEGIN JTABINX := NEXTJTAB;
2154  13   10:8   51                  IF NEXTJTAB = MAXJTAB THEN ERROR(253)
2155  13   10:8   61                  ELSE NEXTJTAB := NEXTJTAB + 1;
2156  13   10:8   74                  JTAB[JTABINX] := OCCURIC
2157  13   10:7   81                END;
2158  13   10:6   84              DISP := -JTABINX;
2159  13   10:6   89              GENBYTE(248-JTABINX-JTABINX)
2160  13   10:5   98            END;
2161  13   10:3   01        END
2162  13   10:2   01      ELSE
2163  13   10:3   03        BEGIN MOVELEFT(REFLIST,CODEP^[IC],2);
2164  13   10:4   12          IF FOP = 57(*UJP*) THEN DISP := IC + 4096
2165  13   10:4   18          ELSE DISP := IC;
2166  13   10:4   29          REFLIST := DISP; IC := IC+2
2167  13   10:3   35        END;
```

```
2168  13   10:0    39   END (*GENJMP*) ;
2169  13   10:0    52
2170  13   11:0     1   PROCEDURE LOAD; FORWARD;
2171  13   11:0     1
2172  13   12:0     1   PROCEDURE GENFJP(FLBP: LBP);
2173  13   12:0     0   BEGIN LOAD;
2174  13   12:1     2     IF GATTR.TYPTR <> BOOLPTR THEN ERROR(135);
2175  13   12:1    14     GENJMP(33(*FJP*),FLBP)
2176  13   12:0    16   END (*GENFJP*) ;
2177  13   12:0    30
2178  13   13:0     1   PROCEDURE GENLABEL(VAR FLBP: LBP);
2179  13   13:0     0   BEGIN NEW(FLBP);
2180  13   13:1     4     WITH FLBP^ DO
2181  13   13:2     8       BEGIN DEFINED := FALSE; REFLIST := MAXADDR END
2182  13   13:0    18   END (*GENLABEL*) ;
2183  13   13:0    30
2184  13   14:0     1   PROCEDURE PUTLABEL(FLBP: LBP);
2185  13   14:0     2     VAR LREF: INTEGER; LOP: OPRANGE;
2186  13   14:0     0   BEGIN
2187  13   14:1     0     WITH FLBP^ DO
2188  13   14:2     3       BEGIN LREF := REFLIST;
2189  13   14:3     7         DEFINED := TRUE; OCCURIC := IC; JTABINX := 0;
2190  13   14:3    20         WHILE LREF < MAXADDR DO
2191  13   14:4    27           BEGIN
2192  13   14:5    27             IF LREF >= 4096 THEN
2193  13   14:6    34               BEGIN LREF := LREF - 4096; LOP := 57(*UJP*) END
2194  13   14:5    44             ELSE LOP := 33(*FJP*);
2195  13   14:5    49             IC := LREF;
2196  13   14:5    52             MOVELEFT(CODEP^[IC],LREF,2);
2197  13   14:5    60             GENJMP(LOP,FLBP)
2198  13   14:4    62           END;
2199  13   14:3    66         IC := OCCURIC
2200  13   14:2    66       END
2201  13   14:0    70   END (*PUTLABEL*) ;
2202  13   14:0    84
2203  13   11:0     1   PROCEDURE LOAD;
2204  13   11:0     1   VAR J,M: INTEGER;
2205  13   11:0     0   BEGIN
2206  13   11:1     0     WITH GATTR DO
2207  13   11:2     0       IF TYPTR <> NIL THEN
2208  13   11:3     5         BEGIN
```

```
2209  13   11:4    5            CASE KIND OF
2210  13   11:4    8              CST:    IF TYPTR^.FORM = LONGINT THEN
2211  13   11:6   14                       WITH GATTR.CVAL.VALP^ DO
2212  13   11:7   17                         BEGIN
2213  13   11:8   17                           M := 10000;
2214  13   11:8   22                           GENLDC(LONGVAL[1]); GENLDC(18(*DCVT*)); GENNR(DECOPS);
2215  13   11:8   39                           FOR J := 2 TO LLENG DO
2216  13   11:9   51                             BEGIN
2217  13   11:0   51                               IF J = LLENG THEN M := TRUNC(PWROFTEN(LLAST));
2218  13   11:0   65                               GENLDC(M); GENLDC(18(*DCVT*)); GENNR(DECOPS);
2219  13   11:0   74                               GENLDC(8(*DMP*)); GENNR(DECOPS);
2220  13   11:0   80                               GENLDC(LONGVAL[J]);
2221  13   11:0   91                               GENLDC(18(*DCVT*)); GENNR(DECOPS);
2222  13   11:0   97                               GENLDC(2(*DAD*)); GENNR(DECOPS)
2223  13   11:9   01                             END
2224  13   11:7   03                         END
2225  13   11:5   10                    ELSE
2226  13   11:6   12                      IF (TYPTR^.FORM = SCALAR) AND (TYPTR <> REALPTR) THEN
2227  13   11:7   23                        GENLDC(CVAL.IVAL)
2228  13   11:6   24                      ELSE
2229  13   11:7   28                        IF TYPTR = NILPTR THEN GEN0(31(*LDCN*))
2230  13   11:7   35                        ELSE
2231  13   11:8   39                          IF TYPTR = REALPTR THEN GEN1(51(*LDC*),2)
2232  13   11:8   47                          ELSE GEN1(51(*LDC*),5);
2233  13   11:4   57              VARBL: CASE ACCESS OF
2234  13   11:5   60                       DRCT:   IF VLEVEL = 1 THEN GEN1(41(*LDO*),DPLMT)
2235  13   11:6   67                               ELSE GEN2(54(*LOD*),LEVEL-VLEVEL,DPLMT);
2236  13   11:5   81                       INDRCT: GEN1(35(*IND*),IDPLMT);
2237  13   11:5   87                       PACKD:  GEN0(58(*LDP*));
2238  13   11:5   92                       MULTI:  GEN1(60(*LDM*),TYPTR^.SIZE);
2239  13   11:5   99                       BYTE:   GEN0(62(*LDB*))
2240  13   11:5   00                     END;
2241  13   11:4   24                EXPR:
2242  13   11:4   24              END;
2243  13   11:4   40              WITH TYPTR^ DO
2244  13   11:5   43                IF ((FORM = POWER) OR
2245  13   11:5   47                   (FORM = LONGINT) AND (KIND <> CST))
2246  13   11:5   56                   AND (KIND <> EXPR) THEN GENLDC(TYPTR^.SIZE);
2247  13   11:4   66              KIND := EXPR
2248  13   11:3   66            END
2249  13   11:0   69    END (*LOAD*) ;
```

```
2250  13    11:0     88
2251  13    15:0      1   PROCEDURE STORE(VAR FATTR: ATTR);
2252  13    15:0      0   BEGIN
2253  13    15:1      0     WITH FATTR DO
2254  13    15:2      3       IF TYPTR <> NIL THEN
2255  13    15:3      9         CASE ACCESS OF
2256  13    15:3     13           DRCT:    IF VLEVEL = 1 THEN GEN1(43(*SRO*),DPLMT)
2257  13    15:4     22                    ELSE GEN2(56(*STR*),LEVEL-VLEVEL,DPLMT);
2258  13    15:3     38           INDRCT:  IF IDPLMT <> 0 THEN ERROR(400)
2259  13    15:4     47                    ELSE GEN0(26(*STO*));
2260  13    15:3     57           PACKD:   GEN0(59(*STP*));
2261  13    15:3     62           MULTI:   GEN1(61(*STM*),TYPTR^.SIZE);
2262  13    15:3     70           BYTE:    GEN0(63(*STB*))
2263  13    15:3     71         END
2264  13    15:0     92   END (*STORE*) ;
2265  13    15:0     04
2266  13    16:0      1   PROCEDURE LOADADDRESS;
2267  13    16:0      0   BEGIN
2268  13    16:1      0     WITH GATTR DO
2269  13    16:2      0       IF TYPTR <> NIL THEN
2270  13    16:3      5         BEGIN
2271  13    16:4      5           CASE KIND OF
2272  13    16:4      8             CST:    IF STRGTYPE(TYPTR) THEN GEN0(38(*LCA*))
2273  13    16:5     17                     ELSE ERROR(400);
2274  13    16:4     29             VARBL:  CASE ACCESS OF
2275  13    16:5     32                       DRCT:    IF VLEVEL = 1 THEN GEN1(37(*LAO*),DPLMT)
2276  13    16:6     39                                ELSE GEN2(50(*LDA*),LEVEL-VLEVEL,DPLMT);
2277  13    16:5     53                       INDRCT:  IF IDPLMT <> 0 THEN GEN1(34(*INC*),IDPLMT);
2278  13    16:5     64                       PACKD:   ERROR(103)
2279  13    16:5     65                     END
2280  13    16:4     84           END;
2281  13    16:4     98           KIND := VARBL; ACCESS := INDRCT; IDPLMT := 0
2282  13    16:3     04         END
2283  13    16:0     07   END (*LOADADDRESS*) ;
2284  13    16:0     20
2285  13    17:0      1   PROCEDURE BYTEADDRESS;
2286  13    17:0      0   BEGIN
2287  13    17:1      0     WITH GATTR DO
2288  13    17:2      0       IF TYPTR <> NIL THEN
2289  13    17:3      5         BEGIN
2290  13    17:4      5           CASE KIND OF
```

```
2291  13   17:4    8              CST:    IF STRGTYPE(TYPTR) THEN GEN0(38(*LCA*))
2292  13   17:5   17                      ELSE ERROR(400);
2293  13   17:4   29              VARBL:  CASE ACCESS OF
2294  13   17:5   32                        DRCT:    IF VLEVEL = 1 THEN GEN1(37(*LAO*),DPLMT)
2295  13   17:6   39                                 ELSE GEN2(50(*LDA*),LEVEL-VLEVEL,DPLMT);
2296  13   17:5   53                        INDRCT:  IF IDPLMT <> 0 THEN GEN1(34(*INC*),IDPLMT);
2297  13   17:5   64                        PACKD:   ERROR(103)
2298  13   17:5   65                      END
2299  13   17:4   84            END;
2300  13   17:4   98            IF KIND <> VARBL THEN BEGIN KIND := VARBL; GENLDC(0) END
2301  13   17:4   09            ELSE
2302  13   17:5   11              IF ACCESS <> BYTE THEN GENLDC(0);
2303  13   17:4   19            ACCESS := BYTE;
2304  13   17:3   22          END
2305  13   17:0   22    END (*BYTEADDRESS*) ;
2306  13   17:0   34
2307  13   18:0    1    PROCEDURE STRGTOPA(FIC: ADDRRANGE);
2308  13   18:0    0      BEGIN
2309  13   18:1    0        IF ODD(FIC) THEN
2310  13   18:2    3          BEGIN
2311  13   18:3    3            MOVERIGHT( CODEP^[FIC+1], CODEP^[FIC+2], ORD(CODEP^[FIC+1])+1 );
2312  13   18:3   20            CODEP^[FIC] := CHR(215(*NOP*)); CODEP^[FIC+1] := CHR(208(*LPA*))
2313  13   18:2   33          END
2314  13   18:1   34        ELSE
2315  13   18:2   36          BEGIN
2316  13   18:3   36            MOVELEFT( CODEP^[FIC+2], CODEP^[FIC+1], ORD(CODEP^[FIC+2])+1 );
2317  13   18:3   53            CODEP^[FIC] := CHR(208); CODEP^[FIC+ORD(CODEP^[FIC+1])+2] := CHR(215)
2318  13   18:2   72          END
2319  13   18:0   73      END (*STRGTOPA*) ;
2320  13   18:0   86
2321  13   19:0    1    PROCEDURE EXPRESSION(FSYS: SETOFSYS); FORWARD;
2322  13   19:0    5
2323  13   20:0    1    PROCEDURE SELECTOR(FSYS: SETOFSYS; FCP: CTP);
2324  13   20:0    6      VAR LATTR: ATTR; LCP: CTP; LMIN,LMAX: INTEGER;
2325  13   20:0    0    BEGIN
2326  13   20:1    0      WITH FCP^, GATTR DO
2327  13   20:2    3        BEGIN TYPTR := IDTYPE; KIND := VARBL;
2328  13   20:3   10          CASE KLASS OF
2329  13   20:3   15            ACTUALVARS:
2330  13   20:4   15              BEGIN VLEVEL := VLEV; DPLMT := VADDR; ACCESS := DRCT;
2331  13   20:5   28                IF INMODULE THEN
```

```
2332  13   20:6   32              IF TYPTR <> NIL THEN
2333  13   20:7   37                IF (VLEV = 1) AND (TYPTR^.FORM = RECORDS) THEN LOADADDRESS
2334  13   20:4   49            END;
2335  13   20:3   53          FORMALVARS:
2336  13   20:4   53            BEGIN
2337  13   20:5   53              IF VLEV = 1 THEN GEN1(41(*LDO*),VADDR)
2338  13   20:5   64              ELSE GEN2(54(*LOD*),LEVEL-VLEV,VADDR);
2339  13   20:5   80              ACCESS := INDRCT; IDPLMT := 0
2340  13   20:4   83            END;
2341  13   20:3   88          FIELD:
2342  13   20:4   88            WITH DISPLAY[DISX] DO
2343  13   20:5   96             BEGIN
2344  13   20:6   96              IF OCCUR = CREC THEN
2345  13   20:7   02                BEGIN ACCESS := DRCT; VLEVEL := CLEV;
2346  13   20:8   09                  DPLMT := CDSPL + FLDADDR
2347  13   20:7   11                END
2348  13   20:6   17              ELSE
2349  13   20:7   19                BEGIN
2350  13   20:8   19                  IF LEVEL = 1 THEN GEN1(41(*LDO*),VDSPL)
2351  13   20:8   28                  ELSE GEN2(54(*LOD*),0,VDSPL);
2352  13   20:8   38                  ACCESS := INDRCT; IDPLMT := FLDADDR
2353  13   20:7   41                END;
2354  13   20:6   46              IF FISPACKD THEN
2355  13   20:7   51                BEGIN LOADADDRESS;
2356  13   20:8   53                  ACCESS := PACKD;
2357  13   20:8   56                  GENLDC(FLDWIDTH); GENLDC(FLDRBIT)
2358  13   20:7   64                END
2359  13   20:5   66             END;
2360  13   20:3   68          FUNC:
2361  13   20:4   68            IF PFDECKIND <> DECLARED THEN ERROR(150)
2362  13   20:4   78            ELSE
2363  13   20:5   83              IF NOT INSCOPE THEN ERROR(103)
2364  13   20:5   90              ELSE
2365  13   20:6   95                  BEGIN ACCESS := DRCT; VLEVEL := PFLEV + 1;
2366  13   20:7   05                    DPLMT := LCAFTERMARKSTACK
2367  13   20:6   05                  END
2368  13   20:3   08        END (*CASE*);
2369  13   20:3   28        IF TYPTR <> NIL THEN
2370  13   20:4   33          IF (TYPTR^.FORM <= POWER) AND
2371  13   20:4   37             (TYPTR^.SIZE > PTRSIZE) THEN
2372  13   20:5   44            BEGIN LOADADDRESS; ACCESS := MULTI END
```

```
2373  13   20:2   49        END (*WITH*);
2374  13   20:1   49      IF NOT (SY IN SELECTSYS + FSYS) THEN
2375  13   20:2   65        BEGIN ERROR(59); SKIP(SELECTSYS + FSYS) END;
2376  13   20:1   85      WHILE SY IN SELECTSYS DO
2377  13   20:2   94        BEGIN
2378  13   20:3   94    (*[*) IF SY = LBRACK THEN
2379  13   20:4   99            BEGIN
2380  13   20:5   99              REPEAT LATTR := GATTR;
2381  13   20:6   05                WITH LATTR DO
2382  13   20:7   05                  IF TYPTR <> NIL THEN
2383  13   20:8   10                    IF TYPTR^.FORM <> ARRAYS THEN
2384  13   20:9   16                      BEGIN ERROR(138); TYPTR := NIL END;
2385  13   20:6   25                LOADADDRESS;
2386  13   20:6   27                INSYMBOL; EXPRESSION(FSYS + [COMMA,RBRACK]);
2387  13   20:6   48                LOAD;
2388  13   20:6   50                IF GATTR.TYPTR <> NIL THEN
2389  13   20:7   55                  IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(113);
2390  13   20:6   65                IF LATTR.TYPTR <> NIL THEN
2391  13   20:7   70                  WITH LATTR.TYPTR^ DO
2392  13   20:8   73                    BEGIN
2393  13   20:9   73                      IF COMPTYPES(INXTYPE,GATTR.TYPTR) THEN
2394  13   20:0   83                        BEGIN
2395  13   20:1   83                          IF (INXTYPE <> NIL) AND
2396  13   20:1   87                              NOT STRGTYPE(LATTR.TYPTR) THEN
2397  13   20:2   97                            BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
2398  13   20:3   06                              IF RANGECHECK THEN
2399  13   20:4   10                                BEGIN GENLDC(LMIN); GENLDC(LMAX);
2400  13   20:5   16                                  GEN0(8(*CHK*))
2401  13   20:4   17                                END;
2402  13   20:3   19                              IF LMIN <> 0 THEN
2403  13   20:4   24                                BEGIN GENLDC(ABS(LMIN));
2404  13   20:5   28                                  IF LMIN > 0 THEN GEN0(21(*SBI*))
2405  13   20:5   34                                  ELSE GEN0(2(*ADI*))
2406  13   20:4   39                                END
2407  13   20:2   41                            END
2408  13   20:0   41                        END
2409  13   20:9   41                      ELSE ERROR(139);
2410  13   20:9   49                      WITH GATTR DO
2411  13   20:0   49                        BEGIN TYPTR := AELTYPE; KIND := VARBL;
2412  13   20:1   56                          ACCESS := INDRCT; IDPLMT := 0;
2413  13   20:1   62                          IF TYPTR <> NIL THEN
```

```
2414  13   20:2   67                              IF AISPACKD THEN
2415  13   20:3   71                                IF ELWIDTH = 8 THEN
2416  13   20:4   77                                  BEGIN ACCESS := BYTE;
2417  13   20:5   80                                    IF STRGTYPE(LATTR.TYPTR) AND RANGECHECK THEN
2418  13   20:6   91                                      GEN0(27(*IXS*))
2419  13   20:5   92                                    ELSE (*LEAVE BASE-INDEX PAIR*)
2420  13   20:4   96                                  END
2421  13   20:3   96                                ELSE
2422  13   20:4   98                                  BEGIN ACCESS := PACKD;
2423  13   20:5   01                                    GEN2(64(*IXP*),ELSPERWD,ELWIDTH)
2424  13   20:4   06                                  END
2425  13   20:2   08                              ELSE
2426  13   20:3   10                                BEGIN GEN1(36(*IXA*),TYPTR^.SIZE);
2427  13   20:4   15                                  IF (TYPTR^.FORM <= POWER) AND
2428  13   20:4   19                                     (TYPTR^.SIZE > PTRSIZE) THEN
2429  13   20:5   26                                    ACCESS := MULTI
2430  13   20:3   26                                END
2431  13   20:0   29                          END
2432  13   20:8   29                      END
2433  13   20:5   29                UNTIL SY <> COMMA;
2434  13   20:5   34                IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
2435  13   20:4   45              END (*IF SY = LBRACK*)
2436  13   20:3   48            ELSE
2437  13   20:4   50  (*.*)      IF SY = PERIOD THEN
2438  13   20:5   55                BEGIN
2439  13   20:6   55                  WITH GATTR DO
2440  13   20:7   55                    BEGIN
2441  13   20:8   55                      IF TYPTR <> NIL THEN
2442  13   20:9   60                        IF TYPTR^.FORM <> RECORDS THEN
2443  13   20:0   66                          BEGIN ERROR(140); TYPTR := NIL END;
2444  13   20:8   75                      INSYMBOL;
2445  13   20:8   78                      IF SY = IDENT THEN
2446  13   20:9   83                        BEGIN
2447  13   20:0   83                          IF TYPTR <> NIL THEN
2448  13   20:1   88                            BEGIN SEARCHSECTION(TYPTR^.FSTFLD,LCP);
2449  13   20:2   95                              IF LCP = NIL THEN
2450  13   20:3   00                                BEGIN ERROR(152); TYPTR := NIL END
2451  13   20:2   09                              ELSE
2452  13   20:3   11                                WITH LCP^ DO
2453  13   20:4   14                                  BEGIN TYPTR := IDTYPE;
2454  13   20:5   18                                    CASE ACCESS OF
```

```
2455  13   20:5   21                                    DRCT:   DPLMT := DPLMT + FLDADDR;
2456  13   20:5   30                                    INDRCT: IDPLMT := IDPLMT + FLDADDR;
2457  13   20:5   39                                    MULTI,BYTE,
2458  13   20:5   39                                    PACKD:  ERROR(400)
2459  13   20:5   42                                  END (*CASE ACCESS*);
2460  13   20:5   64                                  IF FISPACKD THEN
2461  13   20:6   69                                    BEGIN LOADADDRESS;
2462  13   20:7   71                                      ACCESS := PACKD;
2463  13   20:7   74                                      GENLDC(FLDWIDTH); GENLDC(FLDRBIT)
2464  13   20:6   82                                    END;
2465  13   20:5   84                                  IF TYPTR <> NIL THEN
2466  13   20:6   89                                    IF (TYPTR^.FORM <= POWER) AND
2467  13   20:6   93                                       (TYPTR^.SIZE > PTRSIZE) THEN
2468  13   20:7   00                                      BEGIN LOADADDRESS; ACCESS := MULTI END
2469  13   20:4   05                                END
2470  13   20:1   05                            END;
2471  13   20:0   05                          INSYMBOL
2472  13   20:9   05                        END (*SY = IDENT*)
2473  13   20:8   08                      ELSE ERROR(2)
2474  13   20:7   11                    END (*WITH GATTR*)
2475  13   20:5   14                END (*IF SY = PERIOD*)
2476  13   20:4   14              ELSE
2477  13   20:5   16  (*^*)       BEGIN
2478  13   20:6   16                IF GATTR.TYPTR <> NIL THEN
2479  13   20:7   21                  WITH GATTR,TYPTR^ DO
2480  13   20:8   24                    IF (FORM = POINTER) OR (FORM = FILES) THEN
2481  13   20:9   35                      BEGIN LOAD; KIND := VARBL;
2482  13   20:0   40                        ACCESS := INDRCT; IDPLMT := 0;
2483  13   20:0   46                        IF FORM = POINTER THEN TYPTR := ELTYPE
2484  13   20:0   52                        ELSE
2485  13   20:1   58                          BEGIN TYPTR := FILTYPE;
2486  13   20:2   62                            IF TYPTR = NIL THEN ERROR(399)
2487  13   20:1   70                          END;
2488  13   20:0   73                        IF TYPTR <> NIL THEN
2489  13   20:1   78                          IF (TYPTR^.FORM <= POWER) AND
2490  13   20:1   82                             (TYPTR^.SIZE > PTRSIZE) THEN
2491  13   20:2   89                               ACCESS := MULTI
2492  13   20:9   89                      END
2493  13   20:8   92                    ELSE ERROR(141);
2494  13   20:6   00                INSYMBOL
2495  13   20:5   00              END;
```

```
2496  13   20:3    03          IF NOT (SY IN FSYS + SELECTSYS) THEN
2497  13   20:4    19            BEGIN ERROR(6); SKIP(FSYS + SELECTSYS) END
2498  13   20:2    39        END (*WHILE*)
2499  13   20:0    39    END (*SELECTOR*) ;
2500  13   20:0    72
2501  13   20:0    72  (*$I #5:BODYPART.A.TEXT*)
2501  13   20:0    72  (*$I #5:BODYPART.B.TEXT*)
2502  13   20:0    72
2503  13   20:0    72  (*     COPYRIGHT (C) 1979, REGENTS OF THE         *)
2504  13   20:0    72  (*     UNIVERSITY OF CALIFORNIA, SAN DIEGO        *)
2505  13   20:0    72
2506  13   21:D     1    PROCEDURE CALL(FSYS: SETOFSYS; FCP: CTP);
2507  13   21:D     6      VAR LKEY: 1..43; WASLPARENT: BOOLEAN;
2508  13   21:D     8
2509  13   22:D     1      PROCEDURE VARIABLE(FSYS: SETOFSYS);
2510  13   22:D     5        VAR LCP: CTP;
2511  13   22:0     0      BEGIN
2512  13   22:1     0        IF SY = IDENT THEN
2513  13   22:2     5          BEGIN SEARCHID(VARS+[FIELD],LCP); INSYMBOL END
2514  13   22:1    21        ELSE BEGIN ERROR(2); LCP := UVARPTR END;
2515  13   22:1    31        SELECTOR(FSYS,LCP)
2516  13   22:0    39      END (*VARIABLE*) ;
2517  13   22:0    54
2518  13   23:D     1      PROCEDURE STRGVAR(FSYS: SETOFSYS; MUSTBEVAR: BOOLEAN);
2519  13   23:0     0      BEGIN EXPRESSION(FSYS);
2520  13   23:1     9        WITH GATTR DO
2521  13   23:2     9          IF ((KIND = CST) AND (TYPTR = CHARPTR))
2522  13   23:2    17              OR STRGTYPE(TYPTR) THEN
2523  13   23:3    26            IF KIND = VARBL THEN LOADADDRESS
2524  13   23:3    31            ELSE
2525  13   23:4    35              BEGIN
2526  13   23:5    35                IF MUSTBEVAR THEN ERROR(154);
2527  13   23:5    44                IF KIND = CST THEN
2528  13   23:6    49                  BEGIN
2529  13   23:7    49                    IF TYPTR = CHARPTR THEN
2530  13   23:8    55                      BEGIN
2531  13   23:9    55                        WITH SCONST^ DO
2532  13   23:0    59                          BEGIN CCLASS := STRG; SLGTH := 1;
2533  13   23:1    67                            SVAL[1] := CHR(CVAL.IVAL)
2534  13   23:0    74                          END;
2535  13   23:9    75                        CVAL.VALP := SCONST;
```

```
2536  13   23:9    79                    NEW(TYPTR,ARRAYS,TRUE,TRUE);
2537  13   23:9    84                    TYPTR^ := STRGPTR^;
2538  13   23:9    89                    TYPTR^.MAXLENG := 1
2539  13   23:8    92                  END;
2540  13   23:7    94                LOADADDRESS
2541  13   23:6    94              END
2542  13   23:4    96          END
2543  13   23:2    96        ELSE
2544  13   23:3    98          BEGIN
2545  13   23:4    98            IF GATTR.TYPTR <> NIL THEN ERROR(125);
2546  13   23:4    07            GATTR.TYPTR := STRGPTR
2547  13   23:3    07          END
2548  13   23:0    11    END (*STRGVAR*) ;
2549  13   23:0    24
2550  13   24:0     1    PROCEDURE ROUTINE(LKEY: INTEGER);
2551  13   24:0     2
2552  13   25:0     1      PROCEDURE NEWSTMT;
2553  13   25:0     1        LABEL 1;
2554  13   25:0     1        VAR LSP,LSP1: STP; VARTS,LMIN,LMAX: INTEGER;
2555  13   25:0     6            LSIZE,LSZ: ADDRRANGE; LVAL: VALU;
2556  13   25:0     0      BEGIN VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2557  13   25:1    15        LSP := NIL; VARTS := 0; LSIZE := 0;
2558  13   25:1    24        IF GATTR.TYPTR <> NIL THEN
2559  13   25:2    29          WITH GATTR.TYPTR^ DO
2560  13   25:3    32            IF FORM = POINTER THEN
2561  13   25:4    38              BEGIN
2562  13   25:5    38                IF ELTYPE <> NIL THEN
2563  13   25:6    44                  WITH ELTYPE^ DO
2564  13   25:7    48                    BEGIN LSIZE := SIZE;
2565  13   25:8    52                      IF FORM = RECORDS THEN LSP := RECVAR
2566  13   25:7    58                    END
2567  13   25:4    62              END
2568  13   25:3    62            ELSE ERROR(116);
2569  13   25:1    68        WHILE SY = COMMA DO
2570  13   25:2    73          BEGIN INSYMBOL;
2571  13   25:3    76            CONSTANT(FSYS + [COMMA,RPARENT],LSP1,LVAL);
2572  13   25:3    94            VARTS := VARTS + 1;
2573  13   25:3    99            IF LSP = NIL THEN ERROR(158)
2574  13   25:3    07            ELSE
2575  13   25:4    12              IF LSP^.FORM <> TAGFLD THEN ERROR(162)
2576  13   25:4    21              ELSE
```

```
2577  13   25:5    26                IF LSP^.TAGFIELDP <> NIL THEN
2578  13   25:6    32                  IF STRGTYPE(LSP1) OR (LSP1 = REALPTR) THEN ERROR(159)
2579  13   25:6    48                  ELSE
2580  13   25:7    53                    IF COMPTYPES(LSP^.TAGFIELDP^.IDTYPE,LSP1) THEN
2581  13   25:8    64                      BEGIN
2582  13   25:9    64                        LSP1 := LSP^.FSTVAR;
2583  13   25:9    68                        WHILE LSP1 <> NIL DO
2584  13   25:0    73                          WITH LSP1^ DO
2585  13   25:1    76                            IF VARVAL.IVAL = LVAL.IVAL THEN
2586  13   25:2    82                              BEGIN LSIZE := SIZE; LSP := SUBVAR;
2587  13   25:3    90                                GOTO 1
2588  13   25:2    92                              END
2589  13   25:1    92                            ELSE LSP1 := NXTVAR;
2590  13   25:9    00                        LSIZE := LSP^.SIZE; LSP := NIL;
2591  13   25:8    07                      END
2592  13   25:7    07                    ELSE ERROR(116);
2593  13   25:3    13          1:  END (*WHILE*) ;
2594  13   25:1    15            GENLDC(LSIZE);
2595  13   25:1    18            GEN1(30(*CSP*),1(*NEW*))
2596  13   25:0    20          END (*NEWSTMT*) ;
2597  13   25:0    40
2598  13   26:D     1          PROCEDURE MOVE;
2599  13   26:0     0          BEGIN VARIABLE(FSYS + [COMMA]); BYTEADDRESS;
2600  13   26:1    15            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2601  13   26:1    29            IF LKEY = 27 THEN
2602  13   26:2    36              BEGIN EXPRESSION(FSYS + [COMMA]); LOAD END
2603  13   26:1    51            ELSE
2604  13   26:2    53              BEGIN VARIABLE(FSYS + [COMMA]); BYTEADDRESS END;
2605  13   26:1    68            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2606  13   26:1    82            EXPRESSION(FSYS + [RPARENT]); LOAD;
2607  13   26:1    97            IF LKEY = 27 THEN GEN1(30(*CSP*),10(*FLC*))
2608  13   26:1    06            ELSE
2609  13   26:2    10              IF LKEY = 21 THEN GEN1(30(*CSP*),2(*MVL*))
2610  13   26:2    19              ELSE GEN1(30(*CSP*),3(*MVR*))
2611  13   26:0    25          END (*MOVE*) ;
2612  13   26:0    40
2613  13   27:D     1          PROCEDURE EXIT;
2614  13   27:D     1            VAR LCP: CTP;
2615  13   27:0     0          BEGIN
2616  13   27:1     0            IF SY = IDENT THEN
2617  13   27:2     5              BEGIN SEARCHID([PROC,FUNC],LCP); INSYMBOL END
```

```
2618  13   27:1    17          ELSE
2619  13   27:2    19            IF (SY = PROGSY) THEN
2620  13   27:3    24              BEGIN LCP := OUTERBLOCK; INSYMBOL END
2621  13   27:2    31            ELSE LCP := NIL;
2622  13   27:1    36          IF LCP <> NIL THEN
2623  13   27:2    41            IF LCP^.PFDECKIND = DECLARED THEN
2624  13   27:3    48              BEGIN GENLDC(LCP^.PFSEG); GENLDC(LCP^.PFNAME);
2625  13   27:4    58                IF INMODULE THEN
2626  13   27:5    62                  BEGIN LINKERREF(PROC,LCP^.PFSEG,IC-2);
2627  13   27:6    71                    IF SEPPROC THEN LINKERREF(PROC,-LCP^.PFNAME,IC-1);
2628  13   27:5    85                  END
2629  13   27:3    85              END
2630  13   27:2    85            ELSE ERROR(125)
2631  13   27:1    88          ELSE ERROR(125);
2632  13   27:1    97          GEN1(30(*CSP*),4(*XIT*))
2633  13   27:0    99        END (*EXIT*) ;
2634  13   27:0    14
2635  13   28:D     1        PROCEDURE UNITIO;
2636  13   28:0     0        BEGIN
2637  13   28:1     0          IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2638  13   28:1     9          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2639  13   28:1    23          VARIABLE(FSYS + [COMMA]); BYTEADDRESS;
2640  13   28:1    38          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2641  13   28:1    52          EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2642  13   28:1    67          IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2643  13   28:1    76          IF SY = COMMA THEN
2644  13   28:2    81            BEGIN INSYMBOL;
2645  13   28:3    84              IF SY = COMMA THEN GENLDC(0)
2646  13   28:3    90              ELSE
2647  13   28:4    94                BEGIN
2648  13   28:5    94                  EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2649  13   28:5    09                  IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2650  13   28:4    15                END
2651  13   28:2    18            END
2652  13   28:1    18          ELSE GENLDC(0);
2653  13   28:1    23          IF SY = COMMA THEN
2654  13   28:2    28            BEGIN INSYMBOL;
2655  13   28:3    31              EXPRESSION(FSYS + [RPARENT]); LOAD;
2656  13   28:3    46              IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2657  13   28:2    52            END
2658  13   28:1    55          ELSE GENLDC(0);
```

```
2659  13   28:1    60          IF LKEY = 13 THEN GEN1(30(*CSP*),5(*URD*))
2660  13   28:1    69          ELSE GEN1(30(*CSP*),6(*UWT*))
2661  13   28:0    75        END (*UNITIO*);
2662  13   28:0    90
2663  13   29:0     1        PROCEDURE CONCAT;
2664  13   29:0     1          VAR LLC: ADDRRANGE; TEMPLGTH: INTEGER;
2665  13   29:0     0        BEGIN TEMPLGTH := 0;
2666  13   29:1     3          LLC := LC; LC := LC + (STRGLGTH DIV CHRSPERWD) + 1;
2667  13   29:1    17          GENLDC(0); GEN2(56(*STR*),0,LLC);
2668  13   29:1    25          GEN2(50(*LDA*),0,LLC);
2669  13   29:1    30          REPEAT
2670  13   29:2    30            STRGVAR(FSYS + [COMMA,RPARENT],FALSE);
2671  13   29:2    44            TEMPLGTH := TEMPLGTH + GATTR.TYPTR^.MAXLENG;
2672  13   29:2    51            IF TEMPLGTH < STRGLGTH THEN GENLDC(TEMPLGTH)
2673  13   29:2    59            ELSE GENLDC(STRGLGTH);
2674  13   29:2    68            GEN2(77(*CXP*),0(*SYS*),23(*SCONCAT*));
2675  13   29:2    73            GEN2(50(*LDA*),0,LLC);
2676  13   29:2    78            TEST := SY <> COMMA;
2677  13   29:2    83            IF NOT TEST THEN INSYMBOL
2678  13   29:1    87          UNTIL TEST;
2679  13   29:1    93          IF TEMPLGTH < STRGLGTH THEN
2680  13   29:2    00            LC := LLC + (TEMPLGTH DIV CHRSPERWD) + 1
2681  13   29:1    05          ELSE TEMPLGTH := STRGLGTH;
2682  13   29:1    16          IF LC > LCMAX THEN LCMAX := LC;
2683  13   29:1    25          LC := LLC;
2684  13   29:1    28          WITH GATTR DO
2685  13   29:2    28            BEGIN NEW(TYPTR,ARRAYS,TRUE,TRUE);
2686  13   29:3    33              TYPTR^ := STRGPTR^;
2687  13   29:3    38              TYPTR^.MAXLENG := TEMPLGTH
2688  13   29:2    41            END
2689  13   29:0    43        END (*CONCAT*) ;
2690  13   29:0    58
2691  13   30:0     1        PROCEDURE COPYDELETE;
2692  13   30:0     1          VAR LLC: ADDRRANGE; LSP: STP;
2693  13   30:0     0        BEGIN
2694  13   30:1     0          IF LKEY = 19 THEN
2695  13   30:2     7            BEGIN LLC := LC;
2696  13   30:3    10              LC := LC + (STRGLGTH DIV CHRSPERWD) + 1;
2697  13   30:2    21            END;
2698  13   30:1    21          IF LKEY <> 43 THEN
2699  13   30:2    28            BEGIN
```

```
2700  13   30:3    28            STRGVAR(FSYS + [COMMA], LKEY = 18);
2701  13   30:3    46            IF LKEY = 19 THEN
2702  13   30:4    53              BEGIN LSP := GATTR.TYPTR;
2703  13   30:5    56                GEN2(50(*LDA*),0,LLC)
2704  13   30:4    59              END;
2705  13   30:3    61            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2706  13   30:2    75          END;
2707  13   30:1    75        EXPRESSION(FSYS + [COMMA]); LOAD;
2708  13   30:1    90        IF GATTR.TYPTR <> NIL THEN
2709  13   30:2    95          IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2710  13   30:1    04        IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2711  13   30:1    18        EXPRESSION(FSYS + [RPARENT]); LOAD;
2712  13   30:1    33        IF GATTR.TYPTR <> NIL THEN
2713  13   30:2    38          IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2714  13   30:1    47        IF LKEY = 19 THEN
2715  13   30:2    54          BEGIN
2716  13   30:3    54            GEN2(77(*CXP*),0(*SYS*),25(*SCOPY*));
2717  13   30:3    59            GEN2(50(*LDA*),0,LLC);
2718  13   30:3    64            IF LSP^.MAXLENG < STRGLGTH THEN
2719  13   30:4    73              LC := LLC + (LSP^.MAXLENG DIV CHRSPERWD) + 1;
2720  13   30:3    84            IF LC > LCMAX THEN LCMAX := LC;
2721  13   30:3    93            LC := LLC; GATTR.TYPTR := LSP
2722  13   30:2    96          END
2723  13   30:1    99        ELSE
2724  13   30:2    01          IF LKEY = 43 THEN
2725  13   30:3    08            GEN2(77(*CXP*),0(*SYS*),29(*GOTOXY*))
2726  13   30:2    11          ELSE GEN2(77(*CXP*),0(*SYS*),26(*SDELETE*))
2727  13   30:0    18      END (*COPYDELETE*) ;
2728  13   30:0    32
2729  13   31:D     1      PROCEDURE STR;
2730  13   31:0     0      BEGIN
2731  13   31:1     0        WITH GATTR DO
2732  13   31:2     0          BEGIN
2733  13   31:3     0            IF COMPTYPES(LONGINTPTR,TYPTR) THEN
2734  13   31:3    10            ELSE IF TYPTR = INTPTR THEN
2735  13   31:5    17                   BEGIN
2736  13   31:6    17                     GENLDC(18(*DCVT*)); GENNR(DECOPS);
2737  13   31:6    23                     TYPTR := LONGINTPTR
2738  13   31:5    23                   END
2739  13   31:4    27                 ELSE ERROR(125);
2740  13   31:3    33            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
```

```
2741  13   31:3    47              STRGVAR(FSYS + [RPARENT], TRUE);
2742  13   31:3    61              IF STRGTYPE(TYPTR) THEN
2743  13   31:4    69                BEGIN GENLDC(TYPTR^.MAXLENG); GENLDC(12(*DSTR*));
2744  13   31:5    77                  GENNR(DECOPS)
2745  13   31:4    78                END
2746  13   31:3    80              ELSE ERROR(116);
2747  13   31:2    86            END
2748  13   31:0    86        END (*STR*);
2749  13   31:0    98
2750  13   32:0     1        PROCEDURE CLOSE;
2751  13   32:0     0        BEGIN
2752  13   32:1     0          VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2753  13   32:1    15          IF GATTR.TYPTR <> NIL THEN
2754  13   32:2    20            IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125);
2755  13   32:1    30          IF SY = COMMA THEN
2756  13   32:2    35            BEGIN INSYMBOL;
2757  13   32:3    38              IF SY = IDENT THEN
2758  13   32:4    43                BEGIN
2759  13   32:5    43                  IF ID = 'NORMAL  ' THEN GENLDC(0)
2760  13   32:5    62                  ELSE
2761  13   32:6    66                    IF ID = 'LOCK    ' THEN GENLDC(1)
2762  13   32:6    85                    ELSE
2763  13   32:7    89                      IF ID = 'PURGE   ' THEN GENLDC(2)
2764  13   32:7    08                      ELSE
2765  13   32:8    12                        IF ID = 'CRUNCH  ' THEN GENLDC(3)
2766  13   32:8    31                        ELSE ERROR(2);
2767  13   32:5    39                  INSYMBOL
2768  13   32:4    39                END
2769  13   32:3    42              ELSE ERROR(2)
2770  13   32:2    45            END
2771  13   32:1    48          ELSE GENLDC(0);
2772  13   32:1    53          GEN2(77(*CXP*),0(*SYS*),6(*FCLOSE*));
2773  13   32:1    58          IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2774  13   32:0    64        END (*CLOSE*) ;
2775  13   32:0    78
2776  13   33:0     1        PROCEDURE GETPUTETC;
2777  13   33:0     0        BEGIN
2778  13   33:1     0          VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2779  13   33:1    15          IF GATTR.TYPTR <> NIL THEN
2780  13   33:2    20            IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
2781  13   33:2    27            ELSE
```

```
2782  13   33:3   32          IF GATTR.TYPTR^.FILTYPE = NIL THEN ERROR(399);
2783  13   33:1   44        CASE LKEY OF
2784  13   33:1   49          32:  BEGIN
2785  13   33:3   49                 IF SY = COMMA THEN
2786  13   33:4   54                   BEGIN
2787  13   33:5   54                     INSYMBOL; EXPRESSION(FSYS + [RPARENT]); LOAD;
2788  13   33:5   72                     IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2789  13   33:4   78                   END
2790  13   33:3   81                 ELSE ERROR(125);
2791  13   33:3   87                 GENNR(SEEK)
2792  13   33:2   88               END;
2793  13   33:1   92          34:  GEN2(77(*CXP*),0(*SYS*),7(*FGET*));
2794  13   33:1   99          35:  GEN2(77(*CXP*),0(*SYS*),8(*FPUT*));
2795  13   33:1   06          40:  BEGIN
2796  13   33:3   06                 IF GATTR.TYPTR <> NIL THEN
2797  13   33:4   11                   IF GATTR.TYPTR^.FILTYPE <> CHARPTR THEN ERROR(399);
2798  13   33:3   24                 GENLDC(12); GENLDC(0);
2799  13   33:3   30                 GEN2(77(*CXP*),0(*SYS*),17(*WRC*))
2800  13   33:2   33               END
2801  13   33:1   35        END (*CASE*) ;
2802  13   33:1   62        IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2803  13   33:0   68      END (*GETPUTETC*) ;
2804  13   33:0   82
2805  13   34:D    1      PROCEDURE SCAN;
2806  13   34:0    0      BEGIN
2807  13   34:1    0        IF GATTR.TYPTR <> NIL THEN
2808  13   34:2    5          IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2809  13   34:1   14        IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2810  13   34:1   28        IF SY = RELOP THEN
2811  13   34:2   33          BEGIN
2812  13   34:3   33            IF OP = EQOP THEN GENLDC(0)
2813  13   34:3   40            ELSE
2814  13   34:4   44              IF OP = NEOP THEN GENLDC(1)
2815  13   34:4   51              ELSE ERROR(125);
2816  13   34:3   59            INSYMBOL
2817  13   34:2   59          END
2818  13   34:1   62        ELSE ERROR(125);
2819  13   34:1   68        EXPRESSION(FSYS + [COMMA]); LOAD;
2820  13   34:1   83        IF GATTR.TYPTR <> NIL THEN
2821  13   34:2   88          IF GATTR.TYPTR <> CHARPTR THEN ERROR(125);
2822  13   34:1   98        IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
```

```
2823  13   34:1    12          VARIABLE(FSYS + [COMMA,RPARENT]); BYTEADDRESS;
2824  13   34:1    27          IF SY = COMMA THEN
2825  13   34:2    32            BEGIN INSYMBOL;
2826  13   34:3    35              EXPRESSION(FSYS + [RPARENT]); LOAD
2827  13   34:2    48            END
2828  13   34:1    50          ELSE GENLDC(0);
2829  13   34:1    55          GEN1(30(*CSP*),11(*SCN*));
2830  13   34:1    59          GATTR.TYPTR := INTPTR
2831  13   34:0    59        END (*SCAN*) ;
2832  13   34:0    74
2833  13   35:D     1        PROCEDURE BLOCKIO;
2834  13   35:0     0        BEGIN
2835  13   35:1     0          VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2836  13   35:1    15          IF GATTR.TYPTR <> NIL THEN
2837  13   35:2    20            IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
2838  13   35:2    27            ELSE
2839  13   35:3    32              IF GATTR.TYPTR^.FILTYPE <> NIL THEN ERROR(399);
2840  13   35:1    44          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2841  13   35:1    58          VARIABLE(FSYS + [COMMA]); BYTEADDRESS;
2842  13   35:1    73          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2843  13   35:1    87          EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2844  13   35:1    02          IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2845  13   35:1    11          IF SY = COMMA THEN
2846  13   35:2    16            BEGIN INSYMBOL;
2847  13   35:3    19              EXPRESSION(FSYS + [RPARENT]); LOAD;
2848  13   35:3    34              IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2849  13   35:2    40            END
2850  13   35:1    43          ELSE GENLDC(-1);
2851  13   35:1    49          IF LKEY = 37 THEN GENLDC(1) ELSE GENLDC(0);
2852  13   35:1    64          GENLDC(0); GENLDC(0);
2853  13   35:1    70          GEN2(77(*CXP*),0(*SYS*),28(*BLOCKIO*));
2854  13   35:1    75          IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
2855  13   35:1    83          GATTR.TYPTR := INTPTR
2856  13   35:0    83        END (*BLOCKIO*) ;
2857  13   35:0    98
2858  13   36:D     1        PROCEDURE SIZEOF;
2859  13   36:D     1          VAR LCP: CTP;
2860  13   36:0     0        BEGIN
2861  13   36:1     0          IF SY = IDENT THEN
2862  13   36:2     5            BEGIN SEARCHID(VARS + [TYPES,FIELD],LCP); INSYMBOL;
2863  13   36:3    21              IF LCP^.IDTYPE <> NIL THEN
```

```
2864  13   36:4    27                GENLDC(LCP^.IDTYPE^.SIZE*CHRSPERWD)
2865  13   36:2    32              END;
2866  13   36:1    34            GATTR.TYPTR := INTPTR
2867  13   36:0    34          END (*SIZEOF*) ;
2868  13   36:0    50
2869  13   24:0     0    BEGIN (*ROUTINE*)
2870  13   24:1     0      CASE LKEY OF
2871  13   24:1     3        12:       NEWSTMT;
2872  13   24:1     7        13,14:    UNITIO;
2873  13   24:1    11        15:       CONCAT;
2874  13   24:1    15        18,19,43:COPYDELETE;
2875  13   24:1    19        21,22,27:MOVE;
2876  13   24:1    23        23:       EXIT;
2877  13   24:1    27        31:       CLOSE;
2878  13   24:1    31        32,34,
2879  13   24:1    31        35,40:    GETPUTETC;
2880  13   24:1    35        36:       SCAN;
2881  13   24:1    39        37,38:    BLOCKIO;
2882  13   24:1    43        41:       SIZEOF;
2883  13   24:1    47        42:       STR
2884  13   24:1    47      END (*CASES*)
2885  13   24:0    22    END (*ROUTINE*) ;
2886  13   24:0    34
2887  13   24:0    34  (*$I #5:BODYPART.B.TEXT*)
2887  13   24:0    34  (*$I #5:BODYPART.C.TEXT*)
2888  13   24:0    34
2889  13   24:0    34  (*     COPYRIGHT (C) 1979, REGENTS OF THE          *)
2890  13   24:0    34  (*     UNIVERSITY OF CALIFORNIA, SAN DIEGO         *)
2891  13   24:0    34
2892  13   37:D     1      PROCEDURE LOADIDADDR(FCP: CTP);
2893  13   37:0     0      BEGIN
2894  13   37:1     0          WITH FCP^ DO
2895  13   37:2     3            IF KLASS = ACTUALVARS THEN
2896  13   37:3    10              IF VLEV = 1 THEN GEN1(37(*LAO*),VADDR)
2897  13   37:3    21              ELSE GEN2(50(*LDA*),LEVEL-VLEV,VADDR)
2898  13   37:2    35            ELSE  (*FORMALVARS*)
2899  13   37:3    39              IF VLEV = 1 THEN GEN1(41(*LDO*),VADDR)
2900  13   37:3    50              ELSE GEN2(54(*LOD*),LEVEL-VLEV,VADDR)
2901  13   37:0    64      END (*LOADIDADDR*) ;
2902  13   37:0    78
2903  13   38:D     1      PROCEDURE READ;
```

```
2904  13   38:0    1      VAR FILEPTR,LCP: CTP;
2905  13   38:0    0    BEGIN FILEPTR := INPUTPTR;
2906  13   38:1    4      IF (SY = IDENT) AND WASLPARENT THEN
2907  13   38:2   13        BEGIN SEARCHID(VARS+[FIELD],LCP);
2908  13   38:3   26          IF LCP^.IDTYPE <> NIL THEN
2909  13   38:4   32            IF LCP^.IDTYPE^.FORM = FILES THEN
2910  13   38:5   39              IF LCP^.IDTYPE^.FILTYPE = CHARPTR THEN
2911  13   38:6   47                BEGIN INSYMBOL; FILEPTR := LCP;
2912  13   38:7   53                  IF NOT (SY IN [COMMA,RPARENT]) THEN ERROR(20);
2913  13   38:7   64                  IF SY = COMMA THEN INSYMBOL
2914  13   38:6   69                END
2915  13   38:2   72        END
2916  13   38:1   72      ELSE
2917  13   38:2   74        IF WASLPARENT THEN ERROR(2);
2918  13   38:1   83      IF WASLPARENT AND (SY <> RPARENT) THEN
2919  13   38:2   92        BEGIN
2920  13   38:3   92          REPEAT LOADIDADDR(FILEPTR);
2921  13   38:4   95            VARIABLE(FSYS + [COMMA,RPARENT]);
2922  13   38:4   08            IF GATTR.ACCESS = BYTE THEN ERROR(103);
2923  13   38:4   17            LOADADDRESS;
2924  13   38:4   19            IF GATTR.TYPTR <> NIL THEN
2925  13   38:5   24              IF COMPTYPES(INTPTR,GATTR.TYPTR) THEN
2926  13   38:6   33                GEN2(77(*CXP*),0(*SYS*),12(*FRDI*))
2927  13   38:5   36              ELSE
2928  13   38:6   40                IF COMPTYPES(REALPTR,GATTR.TYPTR) THEN
2929  13   38:7   50                  GENNR(FREADREAL)
2930  13   38:6   51                ELSE
2931  13   38:7   55                  IF COMPTYPES(LONGINTPTR,GATTR.TYPTR) THEN
2932  13   38:8   65                    BEGIN GENLDC(GATTR.TYPTR^.SIZE);
2933  13   38:9   69                      GENNR(FREADDEC)
2934  13   38:8   70                    END
2935  13   38:7   72                  ELSE
2936  13   38:8   74                    IF COMPTYPES(CHARPTR,GATTR.TYPTR) THEN
2937  13   38:9   84                      GEN2(77(*CXP*),0(*SYS*),16(*FRDC*))
2938  13   38:8   87                    ELSE
2939  13   38:9   91                      IF STRGTYPE(GATTR.TYPTR) THEN
2940  13   38:0   99                        BEGIN GENLDC(GATTR.TYPTR^.MAXLENG);
2941  13   38:1   04                          GEN2(77(*CXP*),0(*SYS*),18(*FRDS*))
2942  13   38:0   07                        END
2943  13   38:9   09                      ELSE ERROR(125);
2944  13   38:4   15            IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
```

```
2945  13   38:4   23              TEST := SY <> COMMA;
2946  13   38:4   28              IF NOT TEST THEN INSYMBOL
2947  13   38:3   32            UNTIL TEST
2948  13   38:2   35          END;
2949  13   38:1   38        IF LKEY = 2 THEN
2950  13   38:2   45          BEGIN LOADIDADDR(FILEPTR);
2951  13   38:3   48            GEN2(77(*CXP*),0(*SYS*),21(*FRLN*));
2952  13   38:3   53            IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2953  13   38:2   59          END
2954  13   38:0   61    END (*READ*) ;
2955  13   38:0   78
2956  13   39:0    1    PROCEDURE WRITE;
2957  13   39:0    1      VAR LSP: STP; DEFAULT: BOOLEAN;
2958  13   39:0    3          FILEPTR,LCP: CTP; LEN,LMIN,LMAX: INTEGER;
2959  13   39:0    0    BEGIN FILEPTR := OUTPUTPTR;
2960  13   39:1    4      IF (SY = IDENT) AND WASLPARENT THEN
2961  13   39:2   13        BEGIN SEARCHID(VARS + [FIELD,KONST,FUNC],LCP);
2962  13   39:3   26          IF LCP^.IDTYPE <> NIL THEN
2963  13   39:4   32            IF LCP^.IDTYPE^.FORM = FILES THEN
2964  13   39:5   39              IF LCP^.IDTYPE^.FILTYPE = CHARPTR THEN
2965  13   39:6   47                BEGIN INSYMBOL; FILEPTR := LCP;
2966  13   39:7   53                  IF NOT (SY IN [COMMA,RPARENT]) THEN ERROR(20);
2967  13   39:7   64                  IF SY = COMMA THEN INSYMBOL
2968  13   39:6   69                END
2969  13   39:2   72        END;
2970  13   39:1   72      IF WASLPARENT AND (SY <> RPARENT) THEN
2971  13   39:2   81        BEGIN
2972  13   39:3   81          REPEAT LOADIDADDR(FILEPTR);
2973  13   39:4   84            EXPRESSION(FSYS + [COMMA,COLON,RPARENT]);
2974  13   39:4   97            LSP := GATTR.TYPTR;
2975  13   39:4   00            IF LSP <> NIL THEN
2976  13   39:5   05              WITH LSP^ DO
2977  13   39:6   08                BEGIN
2978  13   39:7   08                  IF FORM > LONGINT THEN LOADADDRESS
2979  13   39:7   14                  ELSE
2980  13   39:8   18                    BEGIN LOAD;
2981  13   39:9   20                      IF FORM = LONGINT THEN
2982  13   39:0   26                        BEGIN GENLDC(DECSIZE(MAXDEC)); GENLDC(0(*DAJ*));
2983  13   39:1   37                          GENNR(DECOPS)
2984  13   39:0   38                        END
2985  13   39:8   40                    END
```

```
2986  13   39:6   40              END;
2987  13   39:4   40            IF SY = COLON THEN
2988  13   39:5   45              BEGIN INSYMBOL;
2989  13   39:6   48                EXPRESSION(FSYS + [COMMA,COLON,RPARENT]);
2990  13   39:6   61                IF GATTR.TYPTR <> NIL THEN
2991  13   39:7   66                  IF GATTR.TYPTR <> INTPTR THEN ERROR(20);
2992  13   39:6   75                LOAD; DEFAULT := FALSE
2993  13   39:5   77              END
2994  13   39:4   80            ELSE DEFAULT := TRUE;
2995  13   39:4   85            IF LSP = INTPTR THEN
2996  13   39:5   90              BEGIN IF DEFAULT THEN GENLDC(0);
2997  13   39:6   96                GEN2(77(*CXP*),0(*SYS*),13(*FWRI*))
2998  13   39:5   99              END
2999  13   39:4   01            ELSE
3000  13   39:5   03              IF LSP = REALPTR THEN
3001  13   39:6   09                BEGIN IF DEFAULT THEN GENLDC(0);
3002  13   39:7   15                  IF SY = COLON THEN
3003  13   39:8   20                    BEGIN INSYMBOL;
3004  13   39:9   23                      EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
3005  13   39:9   38                      IF GATTR.TYPTR <> NIL THEN
3006  13   39:0   43                        IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
3007  13   39:8   49                    END
3008  13   39:7   52                  ELSE GENLDC(0);
3009  13   39:7   57                  GENNR(FWRITEREAL)
3010  13   39:6   58                END
3011  13   39:5   60              ELSE
3012  13   39:6   62                IF COMPTYPES(LSP,LONGINTPTR) THEN
3013  13   39:7   72                  BEGIN IF DEFAULT THEN GENLDC(0); GENNR(FWRITEDEC) END
3014  13   39:6   81                ELSE
3015  13   39:7   83                  IF LSP = CHARPTR THEN
3016  13   39:8   89                    BEGIN IF DEFAULT THEN GENLDC(0);
3017  13   39:9   95                      GEN2(77(*CXP*),0(*SYS*),17(*FWRC*))
3018  13   39:8   98                    END
3019  13   39:7   00                  ELSE
3020  13   39:8   02                    IF STRGTYPE(LSP) THEN
3021  13   39:9   10                      BEGIN IF DEFAULT THEN GENLDC(0);
3022  13   39:0   16                        GEN2(77(*CXP*),0(*SYS*),19(*FWRS*))
3023  13   39:9   19                      END
3024  13   39:8   21                    ELSE
3025  13   39:9   23                      IF PAOFCHAR(LSP) THEN
3026  13   39:0   31                        BEGIN LMAX := 0;
```

```
3027  13   39:1   34                          IF LSP^.INXTYPE <> NIL THEN
3028  13   39:2   40                            BEGIN GETBOUNDS(LSP^.INXTYPE,LMIN,LMAX);
3029  13   39:3   49                               LMAX := LMAX - LMIN + 1
3030  13   39:2   52                            END;
3031  13   39:1   56                          IF DEFAULT THEN GENLDC(LMAX);
3032  13   39:1   62                          GENLDC(LMAX);
3033  13   39:1   65                          GEN2(77(*CXP*),0(*SYS*),20(*FWRB*))
3034  13   39:0   68                        END
3035  13   39:9   70                      ELSE ERROR(125);
3036  13   39:4   76                IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
3037  13   39:4   84                TEST := SY <> COMMA;
3038  13   39:4   89                IF NOT TEST THEN INSYMBOL
3039  13   39:3   93              UNTIL TEST;
3040  13   39:2   99            END;
3041  13   39:1   99         IF LKEY = 4 THEN (*WRITELN*)
3042  13   39:2   06           BEGIN LOADIDADDR(FILEPTR);
3043  13   39:3   09             GEN2(77(*CXP*),0(*SYS*),22(*FWLN*));
3044  13   39:3   14             IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
3045  13   39:2   20           END
3046  13   39:0   22     END (*WRITE*) ;
3047  13   39:0   40
3048  13   40:D    1     PROCEDURE CALLNONSPECIAL;
3049  13   40:D    1       LABEL 1;
3050  13   40:D    1       VAR NXT,LCP: CTP; LSP: STP; LB: BOOLEAN;
3051  13   40:D    5           LMIN,LMAX: INTEGER;
3052  13   40:0    0     BEGIN
3053  13   40:1    0       WITH FCP^ DO
3054  13   40:2    5         BEGIN NXT := NEXT;
3055  13   40:3    9           IF PFDECKIND = DECLARED THEN
3056  13   40:4   16             IF PFKIND <> ACTUAL THEN ERROR(400)
3057  13   40:2   26         END;
3058  13   40:1   29       IF SY = LPARENT THEN
3059  13   40:2   34         BEGIN
3060  13   40:3   34           REPEAT
3061  13   40:4   34             IF NXT = NIL THEN ERROR(126);
3062  13   40:4   43             INSYMBOL;
3063  13   40:4   46             EXPRESSION([FSYS + [COMMA,RPARENT]);
3064  13   40:4   59             IF (GATTR.TYPTR <> NIL) AND (NXT <> NIL) THEN
3065  13   40:5   68               BEGIN LSP := NXT^.IDTYPE;
3066  13   40:6   72                 IF (NXT^.KLASS = FORMALVARS) OR (LSP <> NIL) THEN
3067  13   40:7   83                   BEGIN
```

```
3068  13   40:8   83          IF NXT^.KLASS = ACTUALVARS THEN
3069  13   40:9   90            IF GATTR.TYPTR^.FORM <= POWER THEN
3070  13   40:0   96              BEGIN LB := (GATTR.TYPTR = CHARPTR)
3071  13   40:1   00                          AND (GATTR.KIND = CST);
3072  13   40:1   06                LOAD;
3073  13   40:1   08                IF LSP^.FORM = POWER THEN
3074  13   40:2   14                  GEN1(32(*ADJ*),LSP^.SIZE)
3075  13   40:1   17                ELSE
3076  13   40:2   21                IF LSP^.FORM = LONGINT THEN
3077  13   40:3   27                  BEGIN
3078  13   40:4   27                    IF GATTR.TYPTR = INTPTR THEN
3079  13   40:5   32                      BEGIN GENLDC(18(*DCVT*)); GENNR(DECOPS);
3080  13   40:6   38                        GATTR.TYPTR := LONGINTPTR
3081  13   40:5   38                      END;
3082  13   40:4   42                    GENLDC(LSP^.SIZE);
3083  13   40:4   46                    GENLDC(0(*DAJ*));
3084  13   40:4   49                    GENNR(DECOPS)
3085  13   40:3   50                  END
3086  13   40:2   52                ELSE
3087  13   40:3   54                IF (LSP^.FORM = SUBRANGE)
3088  13   40:3   58                      AND RANGECHECK THEN
3089  13   40:4   63                  BEGIN GENLDC(LSP^.MIN.IVAL);
3090  13   40:5   67                    GENLDC(LSP^.MAX.IVAL);
3091  13   40:5   71                    GEN0(8(*CHK*))
3092  13   40:4   72                  END
3093  13   40:3   74                ELSE
3094  13   40:4   76                IF (GATTR.TYPTR = INTPTR) AND
3095  13   40:4   79                      COMPTYPES(LSP,REALPTR) THEN
3096  13   40:5   90                  BEGIN GEN0(10(*FLT*));
3097  13   40:6   93                    GATTR.TYPTR := REALPTR
3098  13   40:5   93                  END
3099  13   40:4   97                ELSE
3100  13   40:5   99                IF LB AND STRGTYPE(LSP) THEN
3101  13   40:6   09                  GATTR.TYPTR := STRGPTR
3102  13   40:0   09              END
3103  13   40:9   13            ELSE (*FORM > POWER*)
3104  13   40:0   15              BEGIN LB := STRGTYPE(GATTR.TYPTR)
3105  13   40:1   16                          AND (GATTR.KIND = CST);
3106  13   40:1   27                LOADADDRESS;
3107  13   40:1   29                IF LB AND PAOFCHAR(LSP) THEN
3108  13   40:2   39                  IF NOT LSP^.AISSTRNG THEN
```

```
3109  13   40:3   44                          BEGIN STRGTOPA(STRGCSTIC);
3110  13   40:4   48                            IF LSP^.INXTYPE <> NIL THEN
3111  13   40:5   54                              BEGIN
3112  13   40:6   54                                GETBOUNDS(LSP^.INXTYPE,LMIN,LMAX);
3113  13   40:6   63                                IF LMAX-LMIN+1 <>
3114  13   40:6   68                                    GATTR.TYPTR^.MAXLENG THEN ERROR(142);
3115  13   40:5   80                              END;
3116  13   40:4   80                            GATTR.TYPTR := LSP
3117  13   40:3   80                          END
3118  13   40:0   83                      END
3119  13   40:8   83                    ELSE (*KLASS = FORMALVARS*)
3120  13   40:9   85                      IF GATTR.KIND = VARBL THEN
3121  13   40:0   90                        BEGIN
3122  13   40:1   90                          IF GATTR.ACCESS = BYTE THEN ERROR(103);
3123  13   40:1   99                          LOADADDRESS;
3124  13   40:1   01                          IF LSP <> NIL THEN
3125  13   40:2   06                            IF LSP^.FORM IN [POWER,LONGINT] THEN
3126  13   40:3   13                              IF GATTR.TYPTR^.SIZE <>
3127  13   40:3   15                                  LSP^.SIZE THEN ERROR(142)
3128  13   40:0   23                        END
3129  13   40:9   26                      ELSE ERROR(154);
3130  13   40:8   34                    IF NOT COMPTYPES(LSP,GATTR.TYPTR) THEN ERROR(142)
3131  13   40:7   47                  END
3132  13   40:5   50              END;
3133  13   40:4   50            IF NXT <> NIL THEN NXT := NXT^.NEXT
3134  13   40:3   56        UNTIL SY <> COMMA;
3135  13   40:3   64        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3136  13   40:2   75      END (*LPARENT*) ;
3137  13   40:1   78    IF NXT <> NIL THEN ERROR(126);
3138  13   40:1   87    WITH FCP^ DO
3139  13   40:2   92      IF PFDECKIND = DECLARED THEN
3140  13   40:3   99        BEGIN
3141  13   40:4   99          IF KLASS = FUNC THEN
3142  13   40:5   06            BEGIN GENLDC(0); GENLDC(0) END;
3143  13   40:4   12          IF INMODULE THEN
3144  13   40:5   16            IF SEPPROC THEN
3145  13   40:6   20              IF (PFSEG = SEG) AND (PFLEV = 1) THEN
3146  13   40:7   33                BEGIN GEN1(79(*CGP*),0); LINKERREF(PROC,-PFNAME,IC-1) END
3147  13   40:6   47              ELSE
3148  13   40:7   49                IF PFLEV = 0 THEN GEN2(77(*CXP*),PFSEG,PFNAME)
3149  13   40:7   63                ELSE ERROR(405) (*CALL NOT ALLOWED IN SEP PROC*)
```

```
3150  13   40:5   70                ELSE
3151  13   40:6   75                  IF IMPORTED THEN
3152  13   40:7   80                    BEGIN GEN2(77(*CXP*),0,PFNAME); LINKERREF(PROC,PFSEG,IC-2) END
3153  13   40:6   96                  ELSE GOTO 1
3154  13   40:4   00              ELSE
3155  13   40:5   02        1:      IF PFSEG <> SEG THEN
3156  13   40:6   09                GEN2(77(*CXP*),PFSEG,PFNAME)
3157  13   40:5   16              ELSE
3158  13   40:6   20                IF PFLEV = 0 THEN GEN1(66(*CBP*),PFNAME)
3159  13   40:6   31                ELSE
3160  13   40:7   35                  IF PFLEV = LEVEL THEN GEN1(78(*CLP*),PFNAME)
3161  13   40:7   47                  ELSE
3162  13   40:8   51                    IF PFLEV = 1 THEN GEN1(79(*CGP*),PFNAME)
3163  13   40:8   62                    ELSE GEN1(46(*CIP*),PFNAME)
3164  13   40:3   70          END
3165  13   40:2   72        ELSE
3166  13   40:3   74          IF CSPNUM = 23 THEN GEN1(30,40)  (* TEMP I.5 TRANSLATION --
3167  13   40:3   83                                              MEM WILL BE CSP 23 IN II.0  *)
3168  13   40:3   83          ELSE
3169  13   40:4   87            IF (CSPNUM <> 21) AND (CSPNUM <> 22) THEN
3170  13   40:5   00              GEN1(30(*CSP*),CSPNUM);
3171  13   40:1   06      GATTR.TYPTR := FCP^.IDTYPE
3172  13   40:0   09    END (*CALLNONSPECIAL*) ;
3173  13   40:0   36
3174  13   21:0    0  BEGIN (*CALL*)
3175  13   21:1    0    IF FCP^.PFDECKIND = SPECIAL THEN
3176  13   21:2    7      BEGIN WASLPARENT := TRUE; LKEY := FCP^.KEY;
3177  13   21:3   15        IF SY = LPARENT THEN INSYMBOL
3178  13   21:3   20        ELSE
3179  13   21:4   25          IF LKEY IN [2,4,5,6] THEN WASLPARENT := FALSE
3180  13   21:4   31          ELSE ERROR(9);
3181  13   21:3   40        IF LKEY IN [7,8,9,10,11,13,14,25,36,39,42] THEN
3182  13   21:4   54          BEGIN EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD END;
3183  13   21:3   68        IF LKEY IN [12,13,14,15,18,19,21,22,23,27,31,32,34,35,36,37,38,
3184  13   21:3   69                    40,41,42,43] THEN ROUTINE(LKEY)
3185  13   21:3   83        ELSE
3186  13   21:4   87          CASE LKEY OF
3187  13   21:4   90             1,2: READ;
3188  13   21:4   94             3,4: WRITE;
3189  13   21:4   98             5,6: BEGIN (*EOF & EOLN*)
3190  13   21:6   98                    IF WASLPARENT THEN
```

```
3191  13    21:7     01          BEGIN VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3192  13    21:8     15            IF GATTR.TYPTR <> NIL THEN
3193  13    21:9     20              IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
3194  13    21:9     27              ELSE
3195  13    21:0     32                IF (GATTR.TYPTR^.FILTYPE <> CHARPTR) AND
3196  13    21:0     37                   (LKEY = 6) THEN ERROR(399)
3197  13    21:7     46            END
3198  13    21:6     49          ELSE
3199  13    21:7     51            LOADIDADDR(INPUTPTR);
3200  13    21:6     55          GENLDC(0); GENLDC(0);
3201  13    21:6     61          IF LKEY = 5 THEN GEN2(77(*CXP*),0(*SYS*),10(*FEOF*))
3202  13    21:6     69          ELSE GEN2(77(*CXP*),0(*SYS*),11(*FEOLN*));
3203  13    21:6     78          GATTR.TYPTR := BOOLPTR
3204  13    21:5     78        END (*EOF*) ;
3205  13    21:4     84  7,8:  BEGIN GENLDC(1); (*PREDSUCC*)
3206  13    21:6     87          IF GATTR.TYPTR <> NIL THEN
3207  13    21:7     92            IF GATTR.TYPTR^.FORM = SCALAR THEN
3208  13    21:8     98              IF LKEY = 8 THEN GEN0(2(*ADI*))
3209  13    21:8     04              ELSE GEN0(21(*SBI*))
3210  13    21:7     09            ELSE ERROR(115)
3211  13    21:5     14        END (*PREDSUCC*) ;
3212  13    21:4     19     9:  BEGIN (*ORD*)
3213  13    21:6     19          IF GATTR.TYPTR <> NIL THEN
3214  13    21:7     24            IF GATTR.TYPTR^.FORM >= POWER THEN ERROR(125);
3215  13    21:6     34          GATTR.TYPTR := INTPTR
3216  13    21:5     34        END (*ORD*) ;
3217  13    21:4     39    10:  BEGIN (*SQR*)
3218  13    21:6     39          IF GATTR.TYPTR <> NIL THEN
3219  13    21:7     44          IF GATTR.TYPTR = INTPTR THEN GEN0(24(*SQI*))
3220  13    21:7     50          ELSE
3221  13    21:8     54            IF GATTR.TYPTR = REALPTR THEN GEN0(25(*SQR*))
3222  13    21:8     61            ELSE BEGIN ERROR(125); GATTR.TYPTR := INTPTR END
3223  13    21:5     72        END (*SQR*) ;
3224  13    21:4     74    11:  BEGIN (*ABS*)
3225  13    21:6     74          IF GATTR.TYPTR <> NIL THEN
3226  13    21:7     79            IF GATTR.TYPTR = INTPTR THEN GEN0(0(*ABI*))
3227  13    21:7     85            ELSE
3228  13    21:8     89              IF GATTR.TYPTR = REALPTR THEN GEN0(1(*ABR*))
3229  13    21:8     96              ELSE BEGIN ERROR(125); GATTR.TYPTR := INTPTR END
3230  13    21:5     07        END (*ABS*) ;
3231  13    21:4     09    16:  BEGIN (*LENGTH*)
```

```
3232  13    21:6    09          STRGVAR(FSYS + [RPARENT],FALSE);
3233  13    21:6    22          GENLDC(0(*INDEX*)); GEN0(62(*LDB*)); GATTR.TYPTR := INTPTR
3234  13    21:5    28        END (*LENGTH*) ;
3235  13    21:4    33    17: BEGIN (*INSERT*)
3236  13    21:6    33          STRGVAR(FSYS + [COMMA],FALSE);
3237  13    21:6    46          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3238  13    21:6    60          STRGVAR(FSYS + [COMMA],TRUE);
3239  13    21:6    73          GENLDC(GATTR.TYPTR^.MAXLENG);
3240  13    21:6    78          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3241  13    21:6    92          EXPRESSION(FSYS + [RPARENT]); LOAD;
3242  13    21:6    06          IF GATTR.TYPTR <> NIL THEN
3243  13    21:7    11            IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
3244  13    21:6    20          GEN2(77(*CXP*),0(*SYS*),24(*SINSERT*))
3245  13    21:5    23        END (*INSERT*) ;
3246  13    21:4    27    20: BEGIN (*POS*)
3247  13    21:6    27          STRGVAR(FSYS + [COMMA],FALSE);
3248  13    21:6    40          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3249  13    21:6    54          STRGVAR(FSYS + [RPARENT],FALSE);
3250  13    21:6    67          GENLDC(0); GENLDC(0);
3251  13    21:6    73          GEN2(77(*CXP*),0(*SYS*),27(*SPOS*));
3252  13    21:6    78          GATTR.TYPTR := INTPTR
3253  13    21:5    78        END (*POS*) ;
3254  13    21:4    83    24: BEGIN (*IDSEARCH*)
3255  13    21:6    83          VARIABLE(FSYS + [COMMA]); LOADADDRESS;
3256  13    21:6    97          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3257  13    21:6    11          VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3258  13    21:6    25          GEN1(30(*CSP*),7(*IDS*))
3259  13    21:5    27        END (*IDSEARCH*) ;
3260  13    21:4    31    25: BEGIN (*TREESEARCH*)
3261  13    21:6    31          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3262  13    21:6    45          VARIABLE(FSYS + [COMMA]); LOADADDRESS;
3263  13    21:6    59          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3264  13    21:6    73          VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3265  13    21:6    87          GATTR.TYPTR := INTPTR;
3266  13    21:6    90          GEN1(30(*CSP*),8(*TRS*))
3267  13    21:5    92        END (*TREESEARCH*) ;
3268  13    21:4    96    26: BEGIN (*TIME*)
3269  13    21:6    96          VARIABLE(FSYS + [COMMA]); LOADADDRESS;
3270  13    21:6    10          IF GATTR.TYPTR <> NIL THEN
3271  13    21:7    15            IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
3272  13    21:6    24          IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
```

```
3273  13    21:6     38                    VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3274  13    21:6     52                    IF GATTR.TYPTR <> NIL THEN
3275  13    21:7     57                      IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
3276  13    21:6     66                    GEN1(30(*CSP*),9(*TIM*))
3277  13    21:5     68                  END (*TIME*) ;
3278  13    21:4     72        33,28,29,30: BEGIN (*OPEN,RESET,REWRITE*)
3279  13    21:6     72                    VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
3280  13    21:6     86                    IF GATTR.TYPTR <> NIL THEN
3281  13    21:7     91                      IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125);
3282  13    21:6     01                    IF SY <> COMMA THEN
3283  13    21:7     06                      IF LKEY = 33 THEN
3284  13    21:8     11                        GEN2(77(*CXP*),0(*SYS*),4(*FRESET*))
3285  13    21:7     14                      ELSE ERROR(20)
3286  13    21:6     19                    ELSE
3287  13    21:7     24                      BEGIN INSYMBOL;
3288  13    21:8     27                        STRGVAR(FSYS + [RPARENT],FALSE);
3289  13    21:8     40                        IF (LKEY = 28) OR (LKEY = 30) THEN
3290  13    21:9     49                          GENLDC(0)
3291  13    21:8     50                        ELSE GENLDC(1);
3292  13    21:8     57                        GENLDC(0); GEN2(77(*CXP*),0(*SYS*),5(*FOPEN*))
3293  13    21:7     63                      END;
3294  13    21:6     65                    IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
3295  13    21:5     71                  END (*OPEN*) ;
3296  13    21:4     75              39: BEGIN (*TRUNC*)
3297  13    21:6     75                    IF GATTR.TYPTR = INTPTR THEN
3298  13    21:7     80                      BEGIN GEN0(10(*FLT*));
3299  13    21:8     83                        GATTR.TYPTR := REALPTR
3300  13    21:7     83                      END;
3301  13    21:6     87                    IF GATTR.TYPTR <> NIL THEN
3302  13    21:7     92                      IF GATTR.TYPTR = REALPTR THEN
3303  13    21:8     98                        GEN1(30(*CSP*),23(*TRUNC*)) (*** TEMPORARY --
3304  13    21:8     00                                                     TRUNC WILL BE CSP 14 IN II.0 ***)
3305  13    21:7     00                      ELSE
3306  13    21:8     04                        IF GATTR.TYPTR^.FORM = LONGINT THEN
3307  13    21:9     10                          BEGIN
3308  13    21:0     10                            GENLDC(20(*DTNC*)); GENNR(DECOPS)
3309  13    21:9     14                          END
3310  13    21:8     16                        ELSE ERROR(125);
3311  13    21:6     22                    GATTR.TYPTR := INTPTR
3312  13    21:5     22                  END
3313  13    21:4     25          END (*SPECIAL CASES*) ;
```

```
3314  13   21:3    12          IF WASLPARENT THEN
3315  13   21:4    15            IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3316  13   21:2    26        END (*SPECIAL PROCEDURES AND FUNCTIONS*)
3317  13   21:1    29      ELSE CALLNONSPECIAL
3318  13   21:0    31    END (*CALL*) ;
3319  13   21:0    54 (*$I #5:BODYPART.C.TEXT*)
3319  13   21:0    54 (*$I #5:BODYPART.D.TEXT*)
3320  13   21:0    54
3321  13   21:0    54 (*    COPYRIGHT (C) 1979, REGENTS OF THE          *)
3322  13   21:0    54 (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO         *)
3323  13   21:0    54
3324  13   19:D     1   PROCEDURE EXPRESSION(*FSYS: SETOFSYS*);
3325  13   19:D     5     LABEL 1;    (* STRING COMPARE KLUDGE *)
3326  13   19:D     5     VAR LATTR: ATTR; LOP: OPERATOR; TYPIND: INTEGER;
3327  13   19:D    12         LSTRGIC,LSIZE: ADDRRANGE; LSTRING,GSTRING: BOOLEAN;
3328  13   19:D    16         LMIN,LMAX: INTEGER;
3329  13   19:D    18
3330  13   41:D     1     PROCEDURE FLOATIT(VAR FSP: STP; FORCEFLOAT: BOOLEAN);
3331  13   41:0     0     BEGIN
3332  13   41:1     0       IF (GATTR.TYPTR = REALPTR) OR (FSP = REALPTR) OR FORCEFLOAT THEN
3333  13   41:2    14         BEGIN
3334  13   41:3    14           IF GATTR.TYPTR = INTPTR THEN
3335  13   41:4    19             BEGIN GEN0(10(*FLT*)); GATTR.TYPTR := REALPTR END;
3336  13   41:3    26           IF FSP = INTPTR THEN
3337  13   41:4    32             BEGIN GEN0(9(*FLO*)); FSP := REALPTR END
3338  13   41:2    39         END
3339  13   41:0    39     END (*FLOATIT*) ;
3340  13   41:0    52
3341  13   42:D     1     PROCEDURE STRETCHIT(VAR FSP: STP);
3342  13   42:0     0     BEGIN
3343  13   42:1     0       IF (FSP^.FORM = LONGINT) OR (GATTR.TYPTR^.FORM = LONGINT) THEN
3344  13   42:2    12         IF GATTR.TYPTR = INTPTR THEN
3345  13   42:3    17           BEGIN GENLDC(18(*DCVT*)); GENNR(DECOPS); GATTR.TYPTR := LONGINTPTR END
3346  13   42:2    27         ELSE
3347  13   42:3    29           IF FSP = INTPTR THEN
3348  13   42:4    35             BEGIN GENLDC(14(*DCV*)); GENNR(DECOPS); FSP := LONGINTPTR END
3349  13   42:0    45     END (*STRETCHIT*) ;
3350  13   42:0    58
3351  13   43:D     1     PROCEDURE SIMPLEEXPRESSION(FSYS: SETOFSYS);
3352  13   43:D     5       VAR LATTR: ATTR; LOP: OPERATOR; SIGNED: BOOLEAN;
3353  13   43:D    12
```

```
3354  13   44:0    1
3355  13   44:0    5
3356  13   44:0   12
3357  13   45:0    1
3358  13   45:0    5
3359  13   45:0    9
3360  13   45:0   14
3361  13   45:0    0
3362  13   45:1    0
3363  13   45:2   10
3364  13   45:3   30
3365  13   45:2   30
3366  13   45:1   33
3367  13   45:2   42
3368  13   45:3   42
3369  13   45:3   45
3370  13   45:4   45
3371  13   45:5   54
3372  13   45:5   57
3373  13   45:6   64
3374  13   45:5   77
3375  13   45:6   79
3376  13   45:7   86
3377  13   45:8   89
3378  13   45:9   97
3379  13   45:8   99
3380  13   45:6   05
3381  13   45:5   17
3382  13   45:6   22
3383  13   45:7   25
3384  13   45:4   32
3385  13   45:3   39
3386  13   45:4   39
3387  13   45:5   39
3388  13   45:6   39
3389  13   45:7   45
3390  13   45:6   47
3391  13   45:5   51
3392  13   45:4   51
3393  13   45:3   56
3394  13   45:4   56
```

```
PROCEDURE TERM(FSYS: SETOFSYS);
  VAR LATTR: ATTR; LSP: STP; LOP: OPERATOR;

  PROCEDURE FACTOR(FSYS: SETOFSYS);
    VAR LCP: CTP; LVP: CSP; VARPART,ALLCONST: BOOLEAN;
        LSP: STP; HIGHVAL,LOWVAL,LIC,LOP: INTEGER;
        CSTPART: SET OF 0..127;
  BEGIN
    IF NOT (SY IN FACBEGSYS) THEN
      BEGIN ERROR(58); SKIP(FSYS + FACBEGSYS);
        GATTR.TYPTR := NIL
      END;
    WHILE SY IN FACBEGSYS DO
      BEGIN
        CASE SY OF
 (*ID*)   IDENT:
            BEGIN SEARCHID([KONST,FORMALVARS,ACTUALVARS,FIELD,FUNC],LCP);
              INSYMBOL;
              IF LCP^.KLASS = FUNC THEN
                BEGIN CALL(FSYS,LCP); GATTR.KIND := EXPR END
              ELSE
                IF LCP^.KLASS = KONST THEN
                  WITH GATTR, LCP^ DO
                    BEGIN TYPTR := IDTYPE; KIND := CST;
                      CVAL := VALUES
                    END
                ELSE SELECTOR(FSYS,LCP);
              IF GATTR.TYPTR <> NIL THEN
                WITH GATTR,TYPTR^ DO
                  IF FORM = SUBRANGE THEN TYPTR := RANGETYPE
            END;
 (*CST*)  INTCONST:
            BEGIN
              WITH GATTR DO
                BEGIN TYPTR := INTPTR; KIND := CST;
                  CVAL := VAL
                END;
              INSYMBOL
            END;
          REALCONST:
            BEGIN
```

```
3395  13   45:5   56                  WITH GATTR DO
3396  13   45:6   56                    BEGIN TYPTR := REALPTR; KIND := CST;
3397  13   45:7   63                      CVAL := VAL
3398  13   45:6   65                    END;
3399  13   45:5   69                  INSYMBOL
3400  13   45:4   69                END;
3401  13   45:3   74              STRINGCONST:
3402  13   45:4   74                BEGIN
3403  13   45:5   74                  WITH GATTR DO
3404  13   45:6   74                    BEGIN
3405  13   45:7   74                      IF LGTH = 1 THEN TYPTR := CHARPTR
3406  13   45:7   80                      ELSE
3407  13   45:8   86                        BEGIN NEW(LSP,ARRAYS,TRUE,TRUE);
3408  13   45:9   91                          LSP^ := STRGPTR^;
3409  13   45:9   96                          LSP^.MAXLENG := LGTH;
3410  13   45:9   02                          TYPTR := LSP
3411  13   45:8   02                        END;
3412  13   45:7   05                      KIND := CST; CVAL := VAL
3413  13   45:6   10                    END;
3414  13   45:5   14                  INSYMBOL
3415  13   45:4   14                END;
3416  13   45:3   19              LONGCONST:
3417  13   45:4   19                BEGIN
3418  13   45:5   19                  WITH GATTR DO
3419  13   45:6   19                    BEGIN NEW(LSP,LONGINT);
3420  13   45:7   24                      LSP^ := LONGINTPTR^;
3421  13   45:7   29                      LSP^.SIZE := DECSIZE(LGTH);
3422  13   45:7   38                      TYPTR := LSP; KIND := CST; CVAL := VAL
3423  13   45:6   46                    END;
3424  13   45:5   50                  INSYMBOL
3425  13   45:4   50                END;
3426  13   45:3   55      (*(*)     LPARENT:
3427  13   45:4   55                BEGIN INSYMBOL; EXPRESSION(FSYS + [RPARENT]);
3428  13   45:5   70                  IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3429  13   45:4   81                END;
3430  13   45:3   86      (*NOT*)   NOTSY:
3431  13   45:4   86                WITH GATTR DO
3432  13   45:5   86                  BEGIN INSYMBOL; FACTOR(FSYS);
3433  13   45:6   98                    IF (KIND = CST) AND (TYPTR = BOOLPTR) THEN
3434  13   45:7   08                      CVAL.IVAL := ORD(NOT ODD(CVAL.IVAL))
3435  13   45:6   10                    ELSE
```

```
3436  13   45:7   14                BEGIN LOAD; GEN0(19(*NOT*));
3437  13   45:8   19                  IF TYPTR <> NIL THEN
3438  13   45:9   24                    IF TYPTR <> BOOLPTR THEN
3439  13   45:0   30                      BEGIN ERROR(135); TYPTR := NIL END
3440  13   45:7   39                  END
3441  13   45:5   39                END;
3442  13   45:3   41        (*[*)   LBRACK:
3443  13   45:4   41                BEGIN INSYMBOL; CSTPART := [ ]; VARPART := FALSE;
3444  13   45:5   54                  NEW(LSP,POWER);
3445  13   45:5   59                  WITH LSP^ DO
3446  13   45:6   62                    BEGIN ELSET := NIL; SIZE := 0; FORM := POWER END;
3447  13   45:5   78                  IF SY = RBRACK THEN
3448  13   45:6   83                    BEGIN
3449  13   45:7   83                      WITH GATTR DO
3450  13   45:8   83                        BEGIN TYPTR := LSP; KIND := CST END;
3451  13   45:7   89                      INSYMBOL
3452  13   45:6   89                    END
3453  13   45:5   92                  ELSE
3454  13   45:6   94                    BEGIN
3455  13   45:7   94                      REPEAT EXPRESSION(FSYS + [COMMA,RBRACK,COLON]);
3456  13   45:8   12                        IF GATTR.TYPTR <> NIL THEN
3457  13   45:9   17                          IF GATTR.TYPTR^.FORM <> SCALAR THEN
3458  13   45:0   23                            BEGIN ERROR(136); GATTR.TYPTR := NIL END
3459  13   45:9   32                          ELSE
3460  13   45:0   34                            IF COMPTYPES(LSP^.ELSET,GATTR.TYPTR) THEN
3461  13   45:1   44                              BEGIN ALLCONST := FALSE; LOP := 23(*SGS*);
3462  13   45:2   50                                IF (GATTR.KIND = CST) AND
3463  13   45:2   53                                   (GATTR.CVAL.IVAL <= 127) THEN
3464  13   45:3   59                                  BEGIN ALLCONST := TRUE;
3465  13   45:4   62                                    LOWVAL := GATTR.CVAL.IVAL;
3466  13   45:4   65                                    HIGHVAL := LOWVAL
3467  13   45:3   65                                  END;
3468  13   45:2   68                                LIC := IC; LOAD;
3469  13   45:2   73                                IF SY = COLON THEN
3470  13   45:3   78                                  BEGIN INSYMBOL; LOP := 20(*SRS*);
3471  13   45:4   84                                    EXPRESSION(FSYS + [COMMA,RBRACK]);
3472  13   45:4   02                                    IF COMPTYPES(LSP^.ELSET,GATTR.TYPTR) THEN
3473  13   45:4   12                                    ELSE
3474  13   45:5   14                                      BEGIN ERROR(137); GATTR.TYPTR:=NIL END;
3475  13   45:4   23                                    IF ALLCONST THEN
3476  13   45:5   26                                      IF (GATTR.KIND = CST) AND
```

```
3477  13   45:5   29                          (GATTR.CVAL.IVAL <= 127) THEN
3478  13   45:6   35                           HIGHVAL := GATTR.CVAL.IVAL
3479  13   45:5   35                        ELSE
3480  13   45:6   40                          BEGIN LOAD; ALLCONST := FALSE END
3481  13   45:4   45                      ELSE LOAD
3482  13   45:3   47                    END;
3483  13   45:2   49                  IF ALLCONST THEN
3484  13   45:3   52                    BEGIN IC := LIC; (*FORGET FIRST CONST*)
3485  13   45:4   55                      CSTPART := CSTPART + [LOWVAL..HIGHVAL]
3486  13   45:3   65                    END
3487  13   45:2   70                  ELSE
3488  13   45:3   72                    BEGIN GEN0(LOP);
3489  13   45:4   75                      IF VARPART THEN GEN0(28(*UNI*))
3490  13   45:4   79                      ELSE VARPART := TRUE
3491  13   45:3   83                    END;
3492  13   45:2   86                  LSP^.ELSET := GATTR.TYPTR;
3493  13   45:2   91                  GATTR.TYPTR := LSP
3494  13   45:1   91                END
3495  13   45:0   94              ELSE ERROR(137);
3496  13   45:8   02            TEST := SY <> COMMA;
3497  13   45:8   07            IF NOT TEST THEN INSYMBOL
3498  13   45:7   11        UNTIL TEST;
3499  13   45:7   17        IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
3500  13   45:6   28      END;
3501  13   45:5   31    IF VARPART THEN
3502  13   45:6   34      BEGIN
3503  13   45:7   34        IF CSTPART <> [ ] THEN
3504  13   45:8   44          BEGIN
3505  13   45:9   44            SCONST^.PVAL := CSTPART;
3506  13   45:9   57            SCONST^.CCLASS := PSET;
3507  13   45:9   61            GATTR.CVAL.VALP := SCONST;
3508  13   45:9   65            GATTR.KIND := CST;
3509  13   45:9   68            LOAD; GEN0(28(*UNI*))
3510  13   45:8   71          END;
3511  13   45:7   73        GATTR.KIND := EXPR
3512  13   45:6   73      END
3513  13   45:5   76    ELSE
3514  13   45:6   78      BEGIN
3515  13   45:7   78        SCONST^.PVAL := CSTPART;
3516  13   45:7   91        SCONST^.CCLASS := PSET;
3517  13   45:7   95        GATTR.CVAL.VALP := SCONST;
```

```
3518  13   45:7   99                        GATTR.KIND := CST
3519  13   45:6   99                      END
3520  13   45:4   02                  END
3521  13   45:3   02              END (*CASE*) ;
3522  13   45:3   10              IF NOT (SY IN FSYS) THEN
3523  13   45:4   20                BEGIN ERROR(6); SKIP(FSYS + FACBEGSYS) END
3524  13   45:2   40          END (*WHILE*)
3525  13   45:0   40      END (*FACTOR*) ;
3526  13   45:0   72
3527  13   44:0    0    BEGIN (*TERM*)
3528  13   44:1    0      FACTOR(FSYS + [MULOP]);
3529  13   44:1   20      WHILE SY = MULOP DO
3530  13   44:2   25        BEGIN LOAD; LATTR := GATTR; LOP := OP;
3531  13   44:3   37          INSYMBOL; FACTOR(FSYS + [MULOP]); LOAD;
3532  13   44:3   62          IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3533  13   44:4   71            CASE LOP OF
3534  13   44:4   74  (***)       MUL:  BEGIN FLOATIT(LATTR.TYPTR,FALSE); STRETCHIT(LATTR.TYPTR);
3535  13   44:6   83                      IF (LATTR.TYPTR = INTPTR) AND (GATTR.TYPTR = INTPTR)
3536  13   44:6   89                        THEN GEN0(15(*MPI*))
3537  13   44:6   93                      ELSE
3538  13   44:7   97                        IF (LATTR.TYPTR = REALPTR) AND
3539  13   44:7   01                           (GATTR.TYPTR = REALPTR) THEN GEN0(16(*MPR*))
3540  13   44:7   09                        ELSE
3541  13   44:8   13                          IF (GATTR.TYPTR^.FORM = LONGINT) AND
3542  13   44:8   17                             (LATTR.TYPTR^.FORM = LONGINT) THEN
3543  13   44:9   24                            BEGIN GENLDC(8(*DMP*)); GENNR(DECOPS) END
3544  13   44:8   30                          ELSE
3545  13   44:9   32                            IF (LATTR.TYPTR^.FORM = POWER)
3546  13   44:9   36                                AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3547  13   44:0   46                              GEN0(12(*INT*))
3548  13   44:9   47                            ELSE BEGIN ERROR(134); GATTR.TYPTR:=NIL END
3549  13   44:5   60                    END;
3550  13   44:4   62  (*/*)       RDIV: BEGIN FLOATIT(LATTR.TYPTR,TRUE);
3551  13   44:6   67                      IF (LATTR.TYPTR = REALPTR) AND
3552  13   44:6   71                         (GATTR.TYPTR = REALPTR) THEN GEN0(7(*DVR*))
3553  13   44:6   79                      ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3554  13   44:5   92                    END;
3555  13   44:4   94  (*DIV*)     IDIV: BEGIN STRETCHIT(LATTR.TYPTR);
3556  13   44:6   98                      IF (LATTR.TYPTR = INTPTR) AND
3557  13   44:6   01                         (GATTR.TYPTR = INTPTR) THEN GEN0(6(*DVI*))
3558  13   44:6   08                      ELSE
```

```
3559  13   44:7    12                          IF (LATTR.TYPTR^.FORM = LONGINT) AND
3560  13   44:7    16                             (GATTR.TYPTR^.FORM = LONGINT) THEN
3561  13   44:6    23                           BEGIN GENLDC(10(*DDV*)); GENNR(DECOPS) END
3562  13   44:7    29                         ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3563  13   44:5    40                     END;
3564  13   44:4    42      (*MOD*)    IMOD: IF (LATTR.TYPTR = INTPTR) AND
3565  13   44:5    45                        (GATTR.TYPTR = INTPTR) THEN GEN0(14(*MOD*))
3566  13   44:5    52                     ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END;
3567  13   44:4    67      (*AND*)    ANDOP:IF (LATTR.TYPTR = BOOLPTR) AND
3568  13   44:5    71                        (GATTR.TYPTR = BOOLPTR) THEN GEN0(4(*AND*))
3569  13   44:5    79                     ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3570  13   44:4    92                END (*CASE*)
3571  13   44:3    12              ELSE GATTR.TYPTR := NIL
3572  13   44:2    14            END (*WHILE*)
3573  13   44:0    17      END (*TERM*) ;
3574  13   44:0    42
3575  13   43:0     0    BEGIN (*SIMPLEEXPRESSION*)
3576  13   43:1     0      SIGNED := FALSE;
3577  13   43:1     3      IF (SY = ADDOP) AND (OP IN [PLUS,MINUS]) THEN
3578  13   43:2    14        BEGIN SIGNED := OP = MINUS; INSYMBOL END;
3579  13   43:1    23      TERM(FSYS + [ADDOP]);
3580  13   43:1    42      IF SIGNED THEN
3581  13   43:2    45        BEGIN LOAD;
3582  13   43:3    47          IF GATTR.TYPTR = INTPTR THEN GEN0(17(*NGI*))
3583  13   43:3    53          ELSE
3584  13   43:4    57            IF GATTR.TYPTR = REALPTR THEN GEN0(18(*NGR*))
3585  13   43:4    64            ELSE
3586  13   43:5    68              IF GATTR.TYPTR^.FORM = LONGINT THEN
3587  13   43:6    74                BEGIN GENLDC(6(*DNG*)); GENNR(DECOPS) END
3588  13   43:5    80              ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3589  13   43:2    91        END;
3590  13   43:1    91      WHILE SY = ADDOP DO
3591  13   43:2    96        BEGIN LOAD; LATTR := GATTR; LOP := OP;
3592  13   43:3    08          INSYMBOL; TERM(FSYS + [ADDOP]); LOAD;
3593  13   43:3    32          IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3594  13   43:4    41            CASE LOP OF
3595  13   43:4    44    (*+*)     PLUS:
3596  13   43:5    44                BEGIN FLOATIT(LATTR.TYPTR,FALSE); STRETCHIT(LATTR.TYPTR);
3597  13   43:6    53                  IF (LATTR.TYPTR = INTPTR)AND(GATTR.TYPTR = INTPTR) THEN
3598  13   43:7    62                    GEN0(2(*ADI*))
3599  13   43:6    63                  ELSE
```

```
3600  13   43:7   67                    IF (LATTR.TYPTR = REALPTR)AND(GATTR.TYPTR = REALPTR) THEN
3601  13   43:8   78                      GEN0(3(*ADR*))
3602  13   43:7   79                    ELSE
3603  13   43:8   83                      IF (GATTR.TYPTR^.FORM = LONGINT) AND
3604  13   43:8   87                         (LATTR.TYPTR^.FORM = LONGINT) THEN
3605  13   43:9   94                        BEGIN GENLDC(2(*DAD*)); GENNR(DECOPS) END
3606  13   43:8   00                      ELSE
3607  13   43:9   02                        IF (LATTR.TYPTR^.FORM = POWER)
3608  13   43:9   06                           AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3609  13   43:0   16                          GEN0(28(*UNI*))
3610  13   43:9   17                        ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3611  13   43:5   30                  END;
3612  13   43:4   32    (*-*)       MINUS:
3613  13   43:5   32                  BEGIN FLOATIT(LATTR.TYPTR,FALSE); STRETCHIT(LATTR.TYPTR);
3614  13   43:6   41                    IF (LATTR.TYPTR = INTPTR) AND (GATTR.TYPTR = INTPTR) THEN
3615  13   43:7   50                      GEN0(21(*SBI*))
3616  13   43:6   51                    ELSE
3617  13   43:7   55                      IF (LATTR.TYPTR = REALPTR) AND (GATTR.TYPTR = REALPTR)
3618  13   43:7   63                        THEN GEN0(22(*SBR*))
3619  13   43:7   67                      ELSE
3620  13   43:8   71                        IF (GATTR.TYPTR^.FORM = LONGINT) AND
3621  13   43:8   75                           (LATTR.TYPTR^.FORM = LONGINT) THEN
3622  13   43:9   82                          BEGIN GENLDC(4(*DSB*)); GENNR(DECOPS) END
3623  13   43:8   88                        ELSE
3624  13   43:9   90                          IF (LATTR.TYPTR^.FORM = POWER)
3625  13   43:9   94                              AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3626  13   43:0   04                            GEN0(5(*DIF*))
3627  13   43:9   05                          ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3628  13   43:5   18                  END;
3629  13   43:4   20    (*OR*)      OROP:
3630  13   43:5   20                  IF (LATTR.TYPTR = BOOLPTR) AND (GATTR.TYPTR = BOOLPTR) THEN
3631  13   43:6   31                    GEN0(13(*IOR*))
3632  13   43:5   32                  ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3633  13   43:4   45              END (*CASE*)
3634  13   43:3   60            ELSE GATTR.TYPTR := NIL
3635  13   43:2   62        END (*WHILE*)
3636  13   43:0   65    END (*SIMPLEEXPRESSION*) ;
3637  13   43:0   90
3638  13   46:D    1    PROCEDURE MAKEPA(VAR STRGFSP: STP; PAFSP: STP);
3639  13   46:D    3      VAR LMIN,LMAX: INTEGER;
3640  13   46:0    0    BEGIN
```

```
3641  13   46:1     0        IF PAFSP^.INXTYPE <> NIL THEN
3642  13   46:2     6          BEGIN GETBOUNDS(PAFSP^.INXTYPE,LMIN,LMAX);
3643  13   46:3    15            IF LMAX-LMIN+1 <> STRGFSP^.MAXLENG THEN ERROR(129)
3644  13   46:2    30          END;
3645  13   46:1    33        STRGFSP := PAFSP
3646  13   46:0    34      END (*MAKEPA*) ;
3647  13   46:0    48
3648  13   19:0     0    BEGIN (*EXPRESSION*)
3649  13   19:1     0      SIMPLEEXPRESSION(FSYS + [RELOP]);
3650  13   19:1    20      IF SY = RELOP THEN
3651  13   19:2    25        BEGIN
3652  13   19:3    25          LSTRING := (GATTR.KIND = CST) AND
3653  13   19:3    28                     (STRGTYPE(GATTR.TYPTR) OR (GATTR.TYPTR = CHARPTR));
3654  13   19:3    42          IF GATTR.TYPTR <> NIL THEN
3655  13   19:4    47            IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3656  13   19:4    53            ELSE LOADADDRESS;
3657  13   19:3    59          LATTR := GATTR; LOP := OP; LSTRGIC := STRGCSTIC;
3658  13   19:3    73          INSYMBOL; SIMPLEEXPRESSION(FSYS);
3659  13   19:3    85          GSTRING := (GATTR.KIND = CST) AND
3660  13   19:3    88                     (STRGTYPE(GATTR.TYPTR) OR (GATTR.TYPTR = CHARPTR));
3661  13   19:3    02          IF GATTR.TYPTR <> NIL THEN
3662  13   19:4    07            IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3663  13   19:4    13            ELSE LOADADDRESS;
3664  13   19:3    19          IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3665  13   19:4    28            IF LOP = INOP THEN
3666  13   19:5    33              IF GATTR.TYPTR^.FORM = POWER THEN
3667  13   19:6    39                IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR^.ELSET) THEN
3668  13   19:7    49                  GEN0(11(*INN*))
3669  13   19:6    50                ELSE BEGIN ERROR(129); GATTR.TYPTR := NIL END
3670  13   19:5    63              ELSE BEGIN ERROR(130); GATTR.TYPTR := NIL END
3671  13   19:4    74            ELSE
3672  13   19:5    76              BEGIN
3673  13   19:6    76                IF LATTR.TYPTR <> GATTR.TYPTR THEN
3674  13   19:7    81                  BEGIN FLOATIT(LATTR.TYPTR,FALSE); STRETCHIT(LATTR.TYPTR) END;
3675  13   19:6    90                IF LSTRING THEN
3676  13   19:7    93                  BEGIN
3677  13   19:8    93                    IF PAOFCHAR(GATTR.TYPTR) THEN
3678  13   19:9    01                      IF NOT GATTR.TYPTR^.AISSTRNG THEN
3679  13   19:0    06                        BEGIN STRGTOPA(LSTRGIC);
3680  13   19:1    09                          MAKEPA(LATTR.TYPTR,GATTR.TYPTR)
3681  13   19:0    12                        END
```

```
3682  13    19:7     14
3683  13    19:6     14
3684  13    19:7     16
3685  13    19:8     19
3686  13    19:9     19
3687  13    19:0     27
3688  13    19:1     32
3689  13    19:2     36
3690  13    19:1     39
3691  13    19:8     41
3692  13    19:6     41
3693  13    19:6     49
3694  13    19:6     62
3695  13    19:7     71
3696  13    19:8     75
3697  13    19:8     79
3698  13    19:9     79
3699  13    19:9     85
3700  13    19:0     90
3701  13    19:0     96
3702  13    19:8     06
3703  13    19:9     06
3704  13    19:0     06
3705  13    19:0     20
3706  13    19:9     20
3707  13    19:8     25
3708  13    19:8     30
3709  13    19:9     30
3710  13    19:0     30
3711  13    19:0     44
3712  13    19:9     44
3713  13    19:8     49
3714  13    19:9     49
3715  13    19:0     49
3716  13    19:0     52
3717  13    19:1     60
3718  13    19:2     64
3719  13    19:1     64
3720  13    19:2     69
3721  13    19:3     72
3722  13    19:4     78
```

```
  END
ELSE
  IF GSTRING THEN
    BEGIN
      IF PAOFCHAR(LATTR.TYPTR) THEN
        IF NOT LATTR.TYPTR^.AISSTRNG THEN
          BEGIN STRGTOPA(STRGCSTIC);
            MAKEPA(GATTR.TYPTR,LATTR.TYPTR)
          END;
    END;
IF (LSTRING AND STRGTYPE(GATTR.TYPTR)) OR
   (GSTRING AND STRGTYPE(LATTR.TYPTR)) THEN GOTO 1;
IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
  BEGIN LSIZE := LATTR.TYPTR^.SIZE; (*INVALID FOR LONG INTEGERS*)
    CASE LATTR.TYPTR^.FORM OF
      SCALAR:
        IF LATTR.TYPTR = REALPTR THEN TYPIND := 1
        ELSE
          IF LATTR.TYPTR = BOOLPTR THEN TYPIND := 3
          ELSE TYPIND := 0;
      POINTER:
        BEGIN
          IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131);
          TYPIND := 0
        END;
      LONGINT: TYPIND := 7;
      POWER:
        BEGIN
          IF LOP IN [LTOP,GTOP] THEN ERROR(132);
          TYPIND := 4
        END;
      ARRAYS:
        BEGIN
          TYPIND := 6;
          IF PAOFCHAR(LATTR.TYPTR) THEN
            IF LATTR.TYPTR^.AISSTRNG THEN
1:            TYPIND := 2
            ELSE
              BEGIN TYPIND := 5;
                IF LATTR.TYPTR^.INXTYPE <> NIL THEN
                  BEGIN
```

```
3723  13    19:5     78                                  GETBOUNDS(LATTR.TYPTR^.INXTYPE,LMIN,LMAX);
3724  13    19:5     87                                  LSIZE := LMAX - LMIN + 1
3725  13    19:4     91                                END
3726  13    19:2     95                              END
3727  13    19:0     95                            ELSE
3728  13    19:1     97                              IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131)
3729  13    19:9     08                          END;
3730  13    19:8     13                        RECORDS:
3731  13    19:9     13                          BEGIN
3732  13    19:0     13                            IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131);
3733  13    19:0     27                            TYPIND := 6
3734  13    19:9     27                          END;
3735  13    19:8     32                        FILES:
3736  13    19:9     32                          BEGIN ERROR(133); TYPIND := 0 END
3737  13    19:8     41                      END;
3738  13    19:8     66                      IF TYPIND = 7 THEN
3739  13    19:9     71                        BEGIN GENLDC(ORD(LOP)); GENLDC(16(*DCMP*));
3740  13    19:0     77                          GENNR(DECOPS)
3741  13    19:9     78                        END
3742  13    19:8     80                      ELSE
3743  13    19:9     82                        CASE LOP OF
3744  13    19:9     85                          LTOP: GEN2(53(*LES*),TYPIND,LSIZE);
3745  13    19:9     92                          LEOP: GEN2(52(*LEQ*),TYPIND,LSIZE);
3746  13    19:9     99                          GTOP: GEN2(49(*GRT*),TYPIND,LSIZE);
3747  13    19:9     06                          GEOP: GEN2(48(*GEQ*),TYPIND,LSIZE);
3748  13    19:9     13                          NEOP: GEN2(55(*NEQ*),TYPIND,LSIZE);
3749  13    19:9     20                          EQOP: GEN2(47(*EQU*),TYPIND,LSIZE)
3750  13    19:9     23                        END
3751  13    19:7     46                    END
3752  13    19:6     46                  ELSE ERROR(129)
3753  13    19:5     51                END;
3754  13    19:3     54              GATTR.TYPTR := BOOLPTR; GATTR.KIND := EXPR
3755  13    19:2     58          END (*SY = RELOP*)
3756  13    19:0     61    END (*EXPRESSION*) ;
3757  13    19:0     86
3758  13    19:0     86 (*$I #5:BODYPART.D.TEXT*)
3758  13    19:0     86 (*$I #5:BODYPART.E.TEXT*)
3759  13    19:0     86
3760  13    19:0     86 (*    COPYRIGHT (C) 1979, REGENTS OF THE          *)
3761  13    19:0     86 (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO         *)
3762  13    19:0     86
```

```
3763  13   47:0    1   PROCEDURE STATEMENT(FSYS: SETOFSYS);
3764  13   47:0    5     LABEL 1;
3765  13   47:0    5     VAR LCP: CTP; TTOP: DISPRANGE; LLP: LABELP; HEAP: ^INTEGER;
3766  13   47:0    9
3767  13   48:0    1     PROCEDURE ASSIGNMENT(FCP: CTP);
3768  13   48:0    2       VAR LATTR: ATTR; CSTRING,PAONLEFT: BOOLEAN; LMIN,LMAX: INTEGER;
3769  13   48:0    0     BEGIN SELECTOR(FSYS + [BECOMES],FCP);
3770  13   48:1   16       IF SY = BECOMES THEN
3771  13   48:2   21         BEGIN LMAX := 0; CSTRING := FALSE;
3772  13   48:3   27           IF GATTR.TYPTR <> NIL THEN
3773  13   48:4   32             IF (GATTR.ACCESS = INDRCT) OR (GATTR.TYPTR^.FORM > POWER) THEN
3774  13   48:5   42               LOADADDRESS;
3775  13   48:3   44           PAONLEFT := PAOFCHAR(GATTR.TYPTR);
3776  13   48:3   52           LATTR := GATTR;
3777  13   48:3   58           INSYMBOL; EXPRESSION(FSYS);
3778  13   48:3   71           IF GATTR.KIND = CST THEN
3779  13   48:4   76             CSTRING := (GATTR.TYPTR = CHARPTR) OR STRGTYPE(GATTR.TYPTR);
3780  13   48:3   89           IF GATTR.TYPTR <> NIL THEN
3781  13   48:4   94             IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3782  13   48:4   00             ELSE LOADADDRESS;
3783  13   48:3   06           IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3784  13   48:4   15             BEGIN
3785  13   48:5   15               IF GATTR.TYPTR = INTPTR THEN
3786  13   48:6   20                 IF COMPTYPES(REALPTR,LATTR.TYPTR) THEN
3787  13   48:7   30                   BEGIN GEN0(10(*FLT*)); GATTR.TYPTR := REALPTR END;
3788  13   48:5   37               IF COMPTYPES(LONGINTPTR,LATTR.TYPTR) THEN
3789  13   48:6   47                 BEGIN
3790  13   48:7   47                   IF GATTR.TYPTR = INTPTR THEN
3791  13   48:8   52                     BEGIN GENLDC(18(*DCVT*)); GENNR(DECOPS);
3792  13   48:9   58                       GATTR.TYPTR := LONGINTPTR
3793  13   48:8   58                     END;
3794  13   48:7   62                   IF GATTR.TYPTR^.FORM <> LONGINT THEN
3795  13   48:8   68                     BEGIN ERROR(129); GATTR.TYPTR := LONGINTPTR END
3796  13   48:6   78                 END;
3797  13   48:5   78               IF PAONLEFT THEN
3798  13   48:6   81                 IF LATTR.TYPTR^.AISSTRNG THEN
3799  13   48:7   85                   IF CSTRING AND (GATTR.TYPTR = CHARPTR) THEN
3800  13   48:8   93                     GATTR.TYPTR := STRGPTR
3801  13   48:7   93                   ELSE
3802  13   48:6   99                 ELSE
3803  13   48:7   01                   IF LATTR.TYPTR^.INXTYPE <> NIL THEN
```

```
3804  13   48:8   07          BEGIN GETBOUNDS(LATTR.TYPTR^.INXTYPE,LMIN,LMAX);
3805  13   48:9   16            LMAX := LMAX - LMIN + 1;
3806  13   48:9   23            IF CSTRING AND (GATTR.TYPTR <> CHARPTR) THEN
3807  13   48:0   31              BEGIN STRGTOPA(STRGCSTIC);
3808  13   48:1   35                IF LMAX <> GATTR.TYPTR^.MAXLENG THEN ERROR(129);
3809  13   48:1   48                GATTR.TYPTR := LATTR.TYPTR
3810  13   48:0   48              END
3811  13   48:8   51          END
3812  13   48:7   51        ELSE GATTR.TYPTR := LATTR.TYPTR;
3813  13   48:5   56    IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3814  13   48:6   65      CASE LATTR.TYPTR^.FORM OF
3815  13   48:6   69        SUBRANGE: BEGIN
3816  13   48:8   69                    IF RANGECHECK THEN
3817  13   48:9   73                      BEGIN
3818  13   48:0   73                        GENLDC(LATTR.TYPTR^.MIN.IVAL);
3819  13   48:0   77                        GENLDC(LATTR.TYPTR^.MAX.IVAL);
3820  13   48:0   81                        GEN0(8(*CHK*))
3821  13   48:9   82                      END;
3822  13   48:8   84                    STORE(LATTR)
3823  13   48:7   86                  END;
3824  13   48:6   90        POWER:    BEGIN
3825  13   48:8   90                  GEN1(32(*ADJ*),LATTR.TYPTR^.SIZE);
3826  13   48:8   95                  STORE(LATTR)
3827  13   48:7   97                  END;
3828  13   48:6   01        SCALAR,
3829  13   48:6   01        POINTER:  STORE(LATTR);
3830  13   48:6   07        LONGINT: BEGIN
3831  13   48:8   07                   GENLDC(LATTR.TYPTR^.SIZE);
3832  13   48:8   11                   GENLDC(0(*DAJ*));
3833  13   48:8   14                   GENNR(DECOPS);
3834  13   48:8   17                   STORE(LATTR)
3835  13   48:7   19                 END;
3836  13   48:6   23        ARRAYS:   IF PAONLEFT THEN
3837  13   48:8   26                   IF LATTR.TYPTR^.AISSTRNG THEN
3838  13   48:9   30                     GEN1(42(*SAS*),LATTR.TYPTR^.MAXLENG)
3839  13   48:8   34                   ELSE GEN1(40(*MOV*),(LMAX+1) DIV 2)
3840  13   48:7   44                 ELSE GEN1(40(*MOV*),LATTR.TYPTR^.SIZE);
3841  13   48:6   55        RECORDS: GEN1(40(*MOV*),LATTR.TYPTR^.SIZE);
3842  13   48:6   62        FILES:   ERROR(146)
3843  13   48:6   65      END
3844  13   48:5   94    ELSE ERROR(129)
```

```
3845  13   48:4   99                END
3846  13   48:2   02              END (*SY = BECOMES*)
3847  13   48:1   02            ELSE ERROR(51)
3848  13   48:0   05        END (*ASSIGNMENT*) ;
3849  13   48:0   26
3850  13   49:0    1        PROCEDURE GOTOSTATEMENT;
3851  13   49:0    1          VAR LLP: LABELP; FOUND: BOOLEAN; TTOP: DISPRANGE;
3852  13   49:0    0        BEGIN
3853  13   49:1    0          IF NOT GOTOOK THEN ERROR(6);
3854  13   49:1    9          IF SY = INTCONST THEN
3855  13   49:2   14            BEGIN
3856  13   49:3   14              FOUND := FALSE; TTOP := TOP;
3857  13   49:3   20              WHILE DISPLAY[TTOP].OCCUR <> BLCK DO TTOP := TTOP - 1;
3858  13   49:3   37              LLP := DISPLAY[TTOP].FLABEL;
3859  13   49:3   45              WHILE (LLP <> NIL) AND NOT FOUND DO
3860  13   49:4   53                WITH LLP^ DO
3861  13   49:5   56                  IF LABVAL = VAL.IVAL THEN
3862  13   49:6   63                    BEGIN FOUND := TRUE;
3863  13   49:7   66                      GENJMP(57(*UJP*),CODELBP)
3864  13   49:6   69                    END
3865  13   49:5   71                  ELSE LLP := NEXTLAB;
3866  13   49:3   79              IF NOT FOUND THEN ERROR(167);
3867  13   49:3   89              INSYMBOL
3868  13   49:2   89            END
3869  13   49:1   92          ELSE ERROR(15)
3870  13   49:0   95        END (*GOTOSTATEMENT*) ;
3871  13   49:0   14
3872  13   50:0    1        PROCEDURE COMPOUNDSTATEMENT;
3873  13   50:0    0        BEGIN
3874  13   50:1    0          REPEAT
3875  13   50:2    0            REPEAT STATEMENT(FSYS + [SEMICOLON,ENDSY])
3876  13   50:2   13            UNTIL NOT (SY IN STATBEGSYS);
3877  13   50:2   25            TEST := SY <> SEMICOLON;
3878  13   50:2   30            IF NOT TEST THEN INSYMBOL
3879  13   50:1   34          UNTIL TEST;
3880  13   50:1   40          IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
3881  13   50:0   51        END (*COMPOUNDSTATEMENET*) ;
3882  13   50:0   70
3883  13   51:0    1        PROCEDURE IFSTATEMENT;
3884  13   51:0    1          VAR LCIX1,LCIX2: LBP; LIC: INTEGER; CONDCOMPILE,NOTHENCLAUSE: BOOLEAN;
3885  13   51:0    0        BEGIN
```

```
3886  13   51:1    0      CONDCOMPILE := FALSE;
3887  13   51:1    3      EXPRESSION(FSYS + [THENSY]);
3888  13   51:1   18      IF (GATTR.KIND = CST) THEN
3889  13   51:2   23        IF (GATTR.TYPTR = BOOLPTR) THEN
3890  13   51:3   29          BEGIN CONDCOMPILE := TRUE;
3891  13   51:4   32            NOTHENCLAUSE := NOT ODD(GATTR.CVAL.IVAL);
3892  13   51:4   36            LIC := IC
3893  13   51:3   36          END;
3894  13   51:1   59      IF NOT CONDCOMPILE THEN
3895  13   51:2   43        BEGIN GENLABEL(LCIX1); GENFJP(LCIX1) END;
3896  13   51:1   50      IF SY = THENSY THEN INSYMBOL ELSE ERROR(52);
3897  13   51:1   64      STATEMENT(FSYS + [ELSESY]);
3898  13   51:1   79      IF CONDCOMPILE THEN
3899  13   51:2   82        IF NOTHENCLAUSE THEN IC := LIC
3900  13   51:2   85        ELSE LIC := IC;
3901  13   51:1   93      IF SY = ELSESY THEN
3902  13   51:2   98        BEGIN
3903  13   51:3   98          IF NOT CONDCOMPILE THEN
3904  13   51:4   02            BEGIN GENLABEL(LCIX2); GENJMP(57(*UJP*),LCIX2); PUTLABEL(LCIX1) END;
3905  13   51:3   13          INSYMBOL; STATEMENT(FSYS);
3906  13   51:3   26          IF CONDCOMPILE THEN
3907  13   51:4   29            BEGIN
3908  13   51:5   29              IF NOT NOTHENCLAUSE THEN IC := LIC
3909  13   51:4   33            END
3910  13   51:3   36          ELSE PUTLABEL(LCIX2)
3911  13   51:2   39        END
3912  13   51:1   41      ELSE
3913  13   51:2   43        IF NOT CONDCOMPILE THEN PUTLABEL(LCIX1)
3914  13   51:0   48    END (*IFSTATEMENT*) ;
3915  13   51:0   62
3916  13   52:D    1    PROCEDURE CASESTATEMENT;
3917  13   52:D    1      LABEL 1;
3918  13   52:D    1      TYPE CIP = ^CASEINFO;
3919  13   52:D    1           CASEINFO = RECORD
3920  13   52:D    1                        NEXT: CIP;
3921  13   52:D    1                        CSSTART: INTEGER;
3922  13   52:D    1                        CSLAB: INTEGER
3923  13   52:D    1                      END;
3924  13   52:D    1      VAR LSP,LSP1: STP; FSTPTR,LPT1,LPT2,LPT3: CIP; LVAL: VALU;
3925  13   52:D    8          LADDR, LCIX: LBP; NULSTMT, LMIN, LMAX: INTEGER;
3926  13   52:0    0    BEGIN EXPRESSION(FSYS + [OFSY,COMMA,COLON]);
```

```
3927  13   52:1   15        LOAD; GENLABEL(LCIX); GENJMP(57(*UJP*),LCIX);
3928  13   52:1   25        LSP := GATTR.TYPTR;
3929  13   52:1   28        IF LSP <> NIL THEN
3930  13   52:2   33          IF (LSP^.FORM <> SCALAR) OR (LSP = REALPTR) THEN
3931  13   52:3   44            BEGIN ERROR(144); LSP := NIL END;
3932  13   52:1   53        IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
3933  13   52:1   67        FSTPTR := NIL; GENLABEL(LADDR);
3934  13   52:1   74        REPEAT
3935  13   52:2   74          LPT3 := NIL;
3936  13   52:2   77          REPEAT CONSTANT(FSYS + [COMMA,COLON],LSP1,LVAL);
3937  13   52:3   95            IF LSP <> NIL THEN
3938  13   52:4   00              IF COMPTYPES(LSP,LSP1) THEN
3939  13   52:5   09                BEGIN LPT1 := FSTPTR; LPT2 := NIL;
3940  13   52:6   15                  WHILE LPT1 <> NIL DO
3941  13   52:7   20                    WITH LPT1^ DO
3942  13   52:8   23                      BEGIN
3943  13   52:9   23                        IF CSLAB <= LVAL.IVAL THEN
3944  13   52:0   29                          BEGIN IF CSLAB = LVAL.IVAL THEN ERROR(156);
3945  13   52:1   41                            GOTO 1
3946  13   52:0   43                          END;
3947  13   52:9   43                        LPT2 := LPT1; LPT1 := NEXT
3948  13   52:8   46                      END;
3949  13   52:6   52    1:              NEW(LPT3);
3950  13   52:6   57                    WITH LPT3^ DO
3951  13   52:7   60                      BEGIN NEXT := LPT1; CSLAB := LVAL.IVAL;
3952  13   52:8   68                        CSSTART := IC
3953  13   52:7   71                      END;
3954  13   52:6   73                    IF LPT2 = NIL THEN FSTPTR := LPT3
3955  13   52:6   78                    ELSE LPT2^.NEXT := LPT3
3956  13   52:5   84                END
3957  13   52:4   86              ELSE ERROR(147);
3958  13   52:3   94            TEST := SY <> COMMA;
3959  13   52:3   99            IF NOT TEST THEN INSYMBOL
3960  13   52:2   03          UNTIL TEST;
3961  13   52:2   09          IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
3962  13   52:2   23          REPEAT STATEMENT(FSYS + [SEMICOLON])
3963  13   52:2   34          UNTIL NOT (SY IN STATBEGSYS);
3964  13   52:2   46          IF LPT3 <> NIL THEN
3965  13   52:3   51            GENJMP(57(*UJP*),LADDR);
3966  13   52:2   55          TEST := SY <> SEMICOLON;
3967  13   52:2   60          IF NOT TEST THEN INSYMBOL
```

```
3968  13   52:1    64      UNTIL TEST OR (SY = ENDSY);
3969  13   52:1    74      PUTLABEL(LCIX);
3970  13   52:1    77      IF FSTPTR <> NIL THEN
3971  13   52:2    32        BEGIN LMAX := FSTPTR^.CSLAB;
3972  13   52:3    86          LPT1 := FSTPTR; FSTPTR := NIL;
3973  13   52:3    92          REPEAT LPT2 := LPT1^.NEXT; LPT1^.NEXT := FSTPTR;
3974  13   52:4    99            FSTPTR := LPT1; LPT1 := LPT2
3975  13   52:3    02          UNTIL LPT1 = NIL;
3976  13   52:3    10          LMIN := FSTPTR^.CSLAB;
3977  13   52:3    14              GEN0(44(*XJP*));
3978  13   52:3    17              GENWORD(LMIN); GENWORD(LMAX);
3979  13   52:3    25              NULSTMT := IC;
3980  13   52:3    28              GENJMP(57(*UJP*),LADDR);
3981  13   52:3    32              REPEAT
3982  13   52:4    32                WITH FSTPTR^ DO
3983  13   52:5    35                  BEGIN
3984  13   52:6    35                    WHILE CSLAB > LMIN DO
3985  13   52:7    41                      BEGIN GENWORD(IC-NULSTMT); LMIN := LMIN + 1 END;
3986  13   52:6    54                    GENWORD(IC-CSSTART);
3987  13   52:6    61                    FSTPTR := NEXT; LMIN := LMIN + 1
3988  13   52:5    66                  END
3989  13   52:3    70              UNTIL FSTPTR = NIL;
3990  13   52:3    75              PUTLABEL(LADDR)
3991  13   52:2    76        END;
3992  13   52:1    78        IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
3993  13   52:0    89    END (*CASESTATEMENT*) ;
3994  13   52:0    18
3995  13   53:D     1    PROCEDURE REPEATSTATEMENT;
3996  13   53:D     1      VAR LADDR: LBP;
3997  13   53:0     0    BEGIN GENLABEL(LADDR); PUTLABEL(LADDR);
3998  13   53:1     7      REPEAT
3999  13   53:2     7        REPEAT STATEMENT(FSYS + [SEMICOLON,UNTILSY])
4000  13   53:2    20        UNTIL NOT (SY IN STATBEGSYS);
4001  13   53:2    32        TEST := SY <> SEMICOLON;
4002  13   53:2    37        IF NOT TEST THEN INSYMBOL
4003  13   53:1    41      UNTIL TEST;
4004  13   53:1    47      IF SY = UNTILSY THEN
4005  13   53:2    52        BEGIN INSYMBOL; EXPRESSION(FSYS); GENFJP(LADDR)
4006  13   53:2    66        END
4007  13   53:1    68      ELSE ERROR(53)
4008  13   53:0    71    END (*REPEATSTATEMENT*) ;
```

```
4009  13   53:0    90
4010  13   54:0     1    PROCEDURE WHILESTATEMENT;
4011  13   54:0     1      VAR LADDR, LCIX: LBP;
4012  13   54:0     0    BEGIN GENLABEL(LADDR); PUTLABEL(LADDR);
4013  13   54:1     7      EXPRESSION(FSYS + [DOSY]); GENLABEL(LCIX); GENFJP(LCIX);
4014  13   54:1    27      IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
4015  13   54:1    41      STATEMENT(FSYS); GENJMP(57(*UJP*),LADDR); PUTLABEL(LCIX)
4016  13   54:0    56    END (*WHILESTATEMENT*) ;
4017  13   54:0    70
4018  13   55:0     1    PROCEDURE FORSTATEMENT;
4019  13   55:0     1      VAR LATTR: ATTR; LSP: STP;  LSY: SYMBOL;
4020  13   55:0     8          LCIX, LADDR: LBP;
4021  13   55:0     0    BEGIN
4022  13   55:1     0      IF SY = IDENT THEN
4023  13   55:2     5        BEGIN SEARCHID(VARS,LCP);
4024  13   55:3    16          WITH LCP^, LATTR DO
4025  13   55:4    21            BEGIN TYPTR := IDTYPE; KIND := VARBL;
4026  13   55:5    28              IF KLASS = ACTUALVARS THEN
4027  13   55:6    35                BEGIN ACCESS := DRCT; VLEVEL := VLEV;
4028  13   55:7    43                  DPLMT := VADDR
4029  13   55:6    43                END
4030  13   55:5    48              ELSE BEGIN ERROR(155); TYPTR := NIL END
4031  13   55:4    59            END;
4032  13   55:3    59          IF LATTR.TYPTR <> NIL THEN
4033  13   55:4    64            IF (LATTR.TYPTR^.FORM > SUBRANGE)
4034  13   55:4    68               OR COMPTYPES(REALPTR,LATTR.TYPTR) THEN
4035  13   55:5    79              BEGIN ERROR(143); LATTR.TYPTR := NIL END;
4036  13   55:3    88          INSYMBOL
4037  13   55:2    88        END
4038  13   55:1    91      ELSE
4039  13   55:2    93        BEGIN ERROR(2); SKIP(FSYS + [BECOMES,TOSY,DOWNTOSY,DOSY])
4040  13   55:2    10        END;
4041  13   55:1    13      IF SY = BECOMES THEN
4042  13   55:2    18        BEGIN INSYMBOL; EXPRESSION(FSYS + [TOSY,DOWNTOSY,DOSY]);
4043  13   55:3    36          IF GATTR.TYPTR <> NIL THEN
4044  13   55:4    41            IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(144)
4045  13   55:4    50              ELSE
4046  13   55:5    55                IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
4047  13   55:6    64                  BEGIN LOAD;
4048  13   55:7    66                    IF LATTR.TYPTR <> NIL THEN
4049  13   55:8    71                      IF (LATTR.TYPTR^.FORM = SUBRANGE) AND RANGECHECK THEN
```

```
4050  13  55:9  80                        BEGIN
4051  13  55:0  80                          GENLDC(LATTR.TYPTR^.MIN.IVAL);
4052  13  55:0  84                          GENLDC(LATTR.TYPTR^.MAX.IVAL);
4053  13  55:0  88                          GEN0(8(*CHK*))
4054  13  55:9  89                        END;
4055  13  55:7  91                    STORE(LATTR)
4056  13  55:6  93                  END
4057  13  55:5  95                ELSE ERROR(145)
4058  13  55:2  00            END
4059  13  55:1  03          ELSE
4060  13  55:2  05            BEGIN ERROR(51); SKIP(FSYS + [TOSY,DOWNTOSY,DOSY]) END;
4061  13  55:1  25          GENLABEL(LADDR);
4062  13  55:1  29          IF SY IN [TOSY,DOWNTOSY] THEN
4063  13  55:2  37            BEGIN LSY := SY; INSYMBOL; EXPRESSION(FSYS + [DOSY]);
4064  13  55:3  56              IF GATTR.TYPTR <> NIL THEN
4065  13  55:4  61                IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(144)
4066  13  55:4  70                ELSE
4067  13  55:5  75                  IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
4068  13  55:6  84                    BEGIN LOAD;
4069  13  55:7  86                      IF LATTR.TYPTR <> NIL THEN
4070  13  55:8  91                        IF (LATTR.TYPTR^.FORM = SUBRANGE) AND RANGECHECK THEN
4071  13  55:9  00                          BEGIN
4072  13  55:0  00                            GENLDC(LATTR.TYPTR^.MIN.IVAL);
4073  13  55:0  04                            GENLDC(LATTR.TYPTR^.MAX.IVAL);
4074  13  55:0  08                            GEN0(8(*CHK*))
4075  13  55:9  09                          END;
4076  13  55:7  11                      GEN2(56(*STR*),0,LC); PUTLABEL(LADDR);
4077  13  55:7  19                      GATTR := LATTR; LOAD; GEN2(54(*LOD*),0,LC);
4078  13  55:7  32                      LC := LC + INTSIZE;
4079  13  55:7  37                      IF LC > LCMAX THEN LCMAX := LC;
4080  13  55:7  46                      IF LSY = TOSY THEN GEN2(52(*LEQ*),0,INTSIZE)
4081  13  55:7  54                      ELSE GEN2(48(*GEQ*),0,INTSIZE);
4082  13  55:6  63                    END
4083  13  55:5  63                  ELSE ERROR(145)
4084  13  55:2  68            END
4085  13  55:1  71          ELSE BEGIN ERROR(55); SKIP(FSYS + [DOSY]) END;
4086  13  55:1  91          GENLABEL(LCIX); GENJMP(33(*FJP*),LCIX);
4087  13  55:1  99          IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
4088  13  55:1  13          STATEMENT(FSYS);
4089  13  55:1  23          GATTR := LATTR; LOAD; GENLDC(1);
4090  13  55:1  34          IF LSY = TOSY THEN GEN0(2(*ADI*)) ELSE GEN0(21(*SBI*));
```

```
4091  13   55:1   47      STORE(LATTR); GENJMP(57(*UJP*),LADDR); PUTLABEL(LCIX);
4092  13   55:1   58      LC := LC - INTSIZE
4093  13   55:0   59    END (*FORSTATEMENT*) ;
4094  13   55:0   73
4095  13   55:0   78
4096  13   56:0    1    PROCEDURE WITHSTATEMENT;
4097  13   56:0    1      VAR LCP: CTP; LCNT1,LCNT2: DISPRANGE;
4098  13   56:0    0    BEGIN LCNT1 := 0; LCNT2 := 0;
4099  13   56:1    6      REPEAT
4100  13   56:2    6        IF SY = IDENT THEN
4101  13   56:3   11          BEGIN SEARCHID(VARS + [FIELD],LCP); INSYMBOL END
4102  13   56:2   27        ELSE BEGIN ERROR(2); LCP := UVARPTR END;
4103  13   56:2   37        SELECTOR(FSYS + [COMMA,DOSY],LCP);
4104  13   56:2   51        IF GATTR.TYPTR <> NIL THEN
4105  13   56:3   56          IF GATTR.TYPTR^.FORM = RECORDS THEN
4106  13   56:4   62            IF TOP < DISPLIMIT THEN
4107  13   56:5   67              BEGIN TOP := TOP + 1; LCNT1 := LCNT1 + 1;
4108  13   56:6   77                WITH DISPLAY[TOP] DO
4109  13   56:7   84                  BEGIN FNAME := GATTR.TYPTR^.FSTFLD END;
4110  13   56:6   88                IF GATTR.ACCESS = DRCT THEN
4111  13   56:7   93                  WITH DISPLAY[TOP] DO
4112  13   56:8   00                    BEGIN OCCUR := CREC; CLEV := GATTR.VLEVEL;
4113  13   56:9   10                      CDSPL := GATTR.DPLMT
4114  13   56:8   13                    END
4115  13   56:6   15                ELSE
4116  13   56:7   17                  BEGIN LOADADDRESS; GEN2(56(*STR*),0,LC);
4117  13   56:8   24                    WITH DISPLAY[TOP] DO
4118  13   56:9   31                      BEGIN OCCUR := VREC; VDSPL := LC END;
4119  13   56:8   41                    LC := LC + PTRSIZE; LCNT2 := LCNT2 + PTRSIZE;
4120  13   56:8   51                    IF LC > LCMAX THEN LCMAX := LC
4121  13   56:7   57                  END
4122  13   56:5   60              END
4123  13   56:4   60            ELSE ERROR(250)
4124  13   56:3   65          ELSE ERROR(140);
4125  13   56:2   76        TEST := SY <> COMMA;
4126  13   56:2   81        IF NOT TEST THEN INSYMBOL
4127  13   56:1   85      UNTIL TEST;
4128  13   56:1   91      IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
4129  13   56:1   05      STATEMENT(FSYS);
4130  13   56:1   15      TOP := TOP - LCNT1; LC := LC - LCNT2;
4131  13   56:0   25    END (*WITHSTATEMENT*) ;
```

```
4132  13   56:0    40
4133  13   47:0     0   BEGIN (*STATEMENT*)
4134  13   47:1     0     STMTLEV := STMTLEV + 1;
4135  13   47:1     6     IF SY = INTCONST THEN (*LABEL*)
4136  13   47:2    11       BEGIN TTOP := TOP;
4137  13   47:3    14         WHILE DISPLAY[TTOP].OCCUR <> BLCK DO TTOP := TTOP-1;
4138  13   47:3    31         LLP := DISPLAY[TTOP].FLABEL;
4139  13   47:3    39         WHILE LLP <> NIL DO
4140  13   47:4    44           WITH LLP^ DO
4141  13   47:5    47             IF LABVAL = VAL.IVAL THEN
4142  13   47:6    54               BEGIN
4143  13   47:7    54                 IF CODELBP^.DEFINED THEN ERROR(165);
4144  13   47:7    65                 PUTLABEL(CODELBP); GOTO 1
4145  13   47:6    71               END
4146  13   47:5    71             ELSE LLP := NEXTLAB;
4147  13   47:3    79         ERROR(167);
4148  13   47:3    85   1:    INSYMBOL;
4149  13   47:3    88         IF SY = COLON THEN INSYMBOL ELSE ERROR(5)
4150  13   47:2    99       END;
4151  13   47:1    02     IF DEBUGGING THEN
4152  13   47:2    06       BEGIN GEN1(85(*BPT*),SCREENDOTS+1); BPTONLINE := TRUE END;
4153  13   47:1    16     IF NOT (SY IN FSYS + [IDENT]) THEN
4154  13   47:2    29       BEGIN ERROR(6); SKIP(FSYS) END;
4155  13   47:1    43     IF SY IN STATBEGSYS + [IDENT] THEN
4156  13   47:2    55       BEGIN MARK(HEAP); (*FOR LABEL CLEANUP*)
4157  13   47:3    59         CASE SY OF
4158  13   47:3    62           IDENT:    BEGIN SEARCHID(VARS + [FIELD,FUNC,PROC],LCP);
4159  13   47:5    75                       INSYMBOL;
4160  13   47:5    78                       IF LCP^.KLASS = PROC THEN CALL(FSYS,LCP)
4161  13   47:5    93                       ELSE ASSIGNMENT(LCP)
4162  13   47:4    98                     END;
4163  13   47:3    02           BEGINSY:  BEGIN INSYMBOL; COMPOUNDSTATEMENT END;
4164  13   47:3    09           GOTOSY:   BEGIN INSYMBOL; GOTOSTATEMENT END;
4165  13   47:3    16           IFSY:     BEGIN INSYMBOL; IFSTATEMENT END;
4166  13   47:3    23           CASESY:   BEGIN INSYMBOL; CASESTATEMENT END;
4167  13   47:3    30           WHILESY:  BEGIN INSYMBOL; WHILESTATEMENT END;
4168  13   47:3    37           REPEATSY: BEGIN INSYMBOL; REPEATSTATEMENT END;
4169  13   47:3    44           FORSY:    BEGIN INSYMBOL; FORSTATEMENT END;
4170  13   47:3    51           WITHSY:   BEGIN INSYMBOL; WITHSTATEMENT END
4171  13   47:3    56         END;
4172  13   47:3    20         RELEASE(HEAP);
```

```
4173  13   47:3    24          IF IC + 100 > MAXCODE THEN
4174  13   47:4    33            BEGIN ERROR(253); IC := 0 END;
4175  13   47:3    42          IF NOT (SY IN [SEMICOLON,ENDSY,ELSESY,UNTILSY]) THEN
4176  13   47:4    51            BEGIN ERROR(6); SKIP(FSYS) END
4177  13   47:2    65        END;
4178  13   47:1    65      STMTLEV := STMTLEV - 1
4179  13   47:0    67    END (*STATEMENT*) ;
4180  13   47:0    90
4181  13   57:0     1  PROCEDURE BODY;
4182  13   57:0     1  VAR LLC1,EXITIC: ADDRRANGE;  LCP: CTP;  LOP: OPRANGE;
4183  13   57:0     5      LLP: LABELP;  LMIN,LMAX: INTEGER;  JTINX: JTABRANGE;
4184  13   57:0     9      DUMMYVAR: ARRAY[0..0] OF INTEGER; (*FOR PRETTY DISPLAY OF STACK AND HEAP*)
4185  13   57:0    10
4186  13   57:0     0  BEGIN
4187  13   57:1     0    IF (NOSWAP) AND (STARTINGUP) THEN
4188  13   57:2     7      BEGIN
4189  13   57:3     7        DECLARATIONPART(FSYS); (* BRING IN DECLARATIONPART *)
4190  13   57:3    18        EXIT(BODYPART);
4191  13   57:2    22      END;
4192  13   57:1    22    NEXTJTAB := 1;
4193  13   57:1    26    IF NOISY THEN
4194  13   57:2    30      BEGIN WRITELN(OUTPUT);
4195  13   57:3    36        IF NOT NOSWAP THEN (*MUST ADJUST DISPLAY OF STACK AND HEAP*)
4196  13   57:4    41          UNITWRITE(3,DUMMYVAR[-1600],35);
4197  13   57:3    56        DUMMYVAR[0]:=MEMAVAIL;
4198  13   57:3    64        IF DUMMYVAR[0] < SMALLESTSPACE THEN SMALLESTSPACE:=DUMMYVAR[0];
4199  13   57:3    83        IF FPROCP <> NIL THEN
4200  13   57:4    90            WRITELN(OUTPUT,FPROCP^.NAME,' [',DUMMYVAR[0]:5,' WORDS]');
4201  13   57:3    49        WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
4202  13   57:2    74      END;
4203  13   57:1    74    IF FPROCP <> NIL THEN
4204  13   57:2    81      BEGIN
4205  13   57:3    81        LLC1 := FPROCP^.LOCALLC; LCP := FPROCP^.NEXT;
4206  13   57:3    94        WHILE LCP <> NIL DO
4207  13   57:4    99          WITH LCP^ DO
4208  13   57:5    02            BEGIN
4209  13   57:6    02                IF IDTYPE <> NIL THEN
4210  13   57:7    08                  IF (KLASS = ACTUALVARS) THEN
4211  13   57:8    15                    IF (IDTYPE^.FORM > POWER) THEN
4212  13   57:9    22                      BEGIN LLC1 := LLC1 - PTRSIZE;
4213  13   57:0    27                        GEN2(50(*LDA*),0,VADDR);
```

```
4214  13   57:0   34                        GEN2(54(*LOD*),0,LLC1);
4215  13   57:0   39                        IF PAOFCHAR(IDTYPE) THEN
4216  13   57:1   48                          WITH IDTYPE^ DO
4217  13   57:2   52                            IF AISSTRNG THEN GEN1(42(*SAS*),MAXLENG)
4218  13   57:2   60                            ELSE
4219  13   57:3   64                              IF INXTYPE <> NIL THEN
4220  13   57:4   70                                BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
4221  13   57:5   79                                  GEN1(40(*MOV*),(LMAX-LMIN+1+1) DIV 2)
4222  13   57:4   89                                END
4223  13   57:3   91                              ELSE
4224  13   57:0   93                        ELSE GEN1(40(*MOV*),IDTYPE^.SIZE)
4225  13   57:9   99                      END
4226  13   57:8   01                    ELSE LLC1 := LLC1 - IDTYPE^.SIZE
4227  13   57:7   06                  ELSE
4228  13   57:8   12                    IF KLASS = FORMALVARS THEN LLC1 := LLC1 - PTRSIZE;
4229  13   57:6   24              LCP := NEXT
4230  13   57:5   24            END;
4231  13   57:2   30    END;
4232  13   57:1   30  STARTDOTS := SCREENDOTS;
4233  13   57:1   34  LCMAX := LC;
4234  13   57:1   37  LLP := DISPLAY[TOP].FLABEL;
4235  13   57:1   45  WHILE LLP <> NIL DO
4236  13   57:2   50    BEGIN GENLABEL(LLP^.CODELBP);
4237  13   57:3   55      LLP := LLP^.NEXTLAB
4238  13   57:2   56    END;
4239  13   57:1   61  IF NOT INMODULE THEN
4240  13   57:2   66    IF LEVEL = 1 THEN
4241  13   57:3   72      BEGIN LCP := USINGLIST;
4242  13   57:4   76        WHILE LCP <> NIL DO
4243  13   57:5   81          BEGIN
4244  13   57:6   81            IF LCP^.SEGID >= 0 THEN
4245  13   57:7   88              BEGIN GENLDC(LCP^.SEGID); GEN1(30(*CSP*),21(*GETSEG*)) END;
4246  13   57:6   97            LCP := LCP^.NEXT
4247  13   57:5   98          END;
4248  13   57:4   03        IF USERINFO.STUPID THEN
4249  13   57:5   07          GEN2(77(*CXP*),6(*TURTLE*),1(*INIT*))
4250  13   57:3   10      END;
4251  13   57:1   12  LCP := DISPLAY[TOP].FFILE;
4252  13   57:1   20  WHILE LCP <> NIL DO
4253  13   57:2   25    WITH LCP^,IDTYPE^ DO
4254  13   57:3   32      BEGIN
```

```
4255  13   57:4   32          GEN2(50(*LDA*),0,VADDR);
4256  13   57:4   39          GEN2(50(*LDA*),0,VADDR+FILESIZE);
4257  13   57:4   50          IF FILTYPE = NIL THEN GENLDC(-1)
4258  13   57:4   58          ELSE
4259  13   57:5   62            IF IDTYPE = INTRACTVPTR THEN GENLDC(0)
4260  13   57:5   70            ELSE
4261  13   57:6   74              IF FILTYPE = CHARPTR THEN GENLDC(-2)
4262  13   57:6   83              ELSE GENLDC(FILTYPE^.SIZE);
4263  13   57:4   92          GEN2(77(*CXP*),0(*SYS*),3(*FINIT*));
4264  13   57:4   97          LCP := NEXT
4265  13   57:3   97        END;
4266  13   57:1   03    IF (LEVEL = 1) AND NOT SYSCOMP THEN
4267  13   57:2   13      GEN1(85(*BPT*),SCREENDOTS+1);
4268  13   57:1   20    REPEAT
4269  13   57:2   20      REPEAT STATEMENT(FSYS + [SEMICOLON,ENDSY])
4270  13   57:2   33      UNTIL NOT (SY IN STATBEGSYS);
4271  13   57:2   45      TEST := SY <> SEMICOLON;
4272  13   57:2   50      IF NOT TEST THEN INSYMBOL
4273  13   57:1   54    UNTIL TEST;
4274  13   57:1   60    IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13);
4275  13   57:1   74    EXITIC := IC;
4276  13   57:1   77    LCP := DISPLAY[TOP].FFILE;
4277  13   57:1   85    WHILE LCP <> NIL DO
4278  13   57:2   90      WITH LCP^ DO
4279  13   57:3   93        BEGIN
4280  13   57:4   93          GEN2(50(*LDA*),0,VADDR);
4281  13   57:4   00          GENLDC(0); GEN2(77(*CXP*),0(*SYS*),6(*FCLOSE*));
4282  13   57:4   08          LCP := NEXT
4283  13   57:3   08        END;
4284  13   57:1   14    IF NOT INMODULE THEN
4285  13   57:2   19      IF LEVEL = 1 THEN
4286  13   57:3   25        BEGIN
4287  13   57:4   25          LCP := USINGLIST;
4288  13   57:4   29          WHILE LCP <> NIL DO
4289  13   57:5   34            BEGIN
4290  13   57:6   34              IF LCP^.SEGID >= 0 THEN
4291  13   57:7   41                BEGIN GENLDC(LCP^.SEGID); GEN1(30(*CSP*),22(*RELSEG*)) END;
4292  13   57:6   50              LCP := LCP^.NEXT
4293  13   57:5   51            END
4294  13   57:3   54        END;
4295  13   57:1   56    IF FPROCP = NIL THEN GEN0(86(*XIT*))
```

```
4296  13   57:1    64   ELSE
4297  13   57:2    68     BEGIN
4298  13   57:3    68       IF FPROCP^.PFLEV = 0 THEN LOP := 65(*RBP*)
4299  13   57:3    77       ELSE LOP := 45(*RNP*);
4300  13   57:3    85       IF FPROCP^.IDTYPE = NIL THEN GEN1(LOP,0)
4301  13   57:3    95       ELSE GEN1(LOP,FPROCP^.IDTYPE^.SIZE)
4302  13   57:2    05     END;
4303  13   57:1    07   LLP := DISPLAY[TOP].FLABEL;  (* CHECK UNDEFINED LABELS *)
4304  13   57:1    15   WHILE LLP <> NIL DO
4305  13   57:2    20     WITH LLP^,CODELBP^ DO
4306  13   57:3    27       BEGIN
4307  13   57:4    27         IF NOT DEFINED THEN
4308  13   57:5    32           IF REFLIST <> MAXADDR THEN ERROR(168);
4309  13   57:4    46         LLP := NEXTLAB
4310  13   57:3    46       END;
4311  13   57:1    52   JTINX := NEXTJTAB - 1;
4312  13   57:1    59   IF ODD(IC) THEN IC := IC + 1;
4313  13   57:1    67   WHILE JTINX > 0 DO
4314  13   57:2    72     BEGIN GENWORD(IC-JTAB[JTINX]); JTINX := JTINX-1 END;
4315  13   57:1    91   IF FPROCP = NIL THEN
4316  13   57:2    98     BEGIN GENWORD((LCMAX-LCAFTERMARKSTACK)*2); GENWORD(0) END
4317  13   57:1    11   ELSE
4318  13   57:2    13     WITH FPROCP^ DO
4319  13   57:3    18       BEGIN GENWORD((LCMAX-LOCALLC)*2);
4320  13   57:4    29         GENWORD((LOCALLC-LCAFTERMARKSTACK)*2)
4321  13   57:3    36       END;
4322  13   57:1    39   GENWORD(IC-EXITIC); GENWORD(IC);
4323  13   57:1    49   GENBYTE(CURPROC); GENBYTE(LEVEL-1);
4324  13   57:1    61   IF NOT CODEINSEG THEN
4325  13   57:2    66     BEGIN CODEINSEG := TRUE;
4326  13   57:3    69       SEGTABLE[SEG].DISKADDR := CURBLK
4327  13   57:2    77     END;
4328  13   57:1    81   WRITECODE(FALSE);
4329  13   57:1    85   SEGINX := SEGINX + IC;
4330  13   57:1    91   PROCTABLE[CURPROC] := SEGINX - 2
4331  13   57:0    00 END (*BODY*) ;
4332  13   57:0    42
4333  13    1:0     0 BEGIN (*BODYPART*)
4334  13    1:1     0   BODY
4335  13    1:0     0 END ;
4336  13    1:0    14
```

```
4337  13    1:0    14
4338  13    1:0    14 (*$I #5:BODYPART.E.TEXT*)
4338  13    1:0    14 (*$I #5:UNITPART.TEXT*)
4339  13    1:0    14
4340  13    1:0    14
4341  13    1:0    14     (*****************************************************************)
4342  13    1:0    14     (*                                                               *)
4343  13    1:0    14     (*  COPYRIGHT (C) 1979 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
4344  13    1:0    14     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
4345  13    1:0    14     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
4346  13    1:0    14     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
4347  13    1:0    14     (*                                                               *)
4348  13    1:0    14     (*****************************************************************)
4349  13    1:0    14
4350  14    1:D     1 SEGMENT PROCEDURE WRITELINKERINFO(DECSTUFF:BOOLEAN);
4351  14    1:D     2   TYPE
4352  14    1:D     2     LITYPES = (EOFMARK,MODDULE,GLOBREF,PUBBLIC,PRIVVATE,CONNSTANT,GLOBDEF,
4353  14    1:D     2                PUBLICDEF,CONSTDEF,EXTPROC,EXTFUNC,SSEPPROC,SSEPFUNC,
4354  14    1:D     2                SEPPREF,SEPFREF);
4355  14    1:D     2     OPFORMAT = (WORD, BYTE, BIG);
4356  14    1:D     2     LIENTRY = RECORD
4357  14    1:D     2                 LINAME: ALPHA;
4358  14    1:D     2                 CASE LITYPE: LITYPES OF
4359  14    1:D     2                   MODDULE,
4360  14    1:D     2                   PUBBLIC,
4361  14    1:D     2                   PRIVVATE,
4362  14    1:D     2                   SEPPREF,
4363  14    1:D     2                   SEPFREF:          (FORMAT: OPFORMAT;
4364  14    1:D     2                                      NREFS: INTEGER;
4365  14    1:D     2                                      NWORDS: INTEGER);
4366  14    1:D     2                   CONSTDEF:         (CONSTANT: INTEGER);
4367  14    1:D     2                   PUBLICDEF:        (BASEOFFSET: INTEGER);
4368  14    1:D     2                   EXTPROC,EXTFUNC,
4369  14    1:D     2                   SSEPPROC,SSEPFUNC:(PROCNUM: INTEGER;
4370  14    1:D     2                                      NPARAMS: INTEGER;
4371  14    1:D     2                                      RANGE: ^INTEGER)
4372  14    1:D     2               END;
4373  14    1:D     2
4374  14    1:D     2   VAR FCP,LCP: CTP; CURRENTBLOCK: INTEGER; I: NONRESIDENT;
4375  14    1:D     6       EXTNAME: ALPHA; FIC: ADDRRANGE;
4376  14    1:D    11       LIREC: LIENTRY;
```

```
4377  14    1:0    19
4378  14    2:0     1   PROCEDURE GETREFS(ID,LENGTH: INTEGER);
4379  14    2:0     3     VAR LIC: ADDRRANGE; J,MAX,BLOCKCOUNT,COUNT: INTEGER;
4380  14    2:0     8
4381  14    3:0     1     PROCEDURE GETNEXTBLOCK;
4382  14    3:0     0     BEGIN
4383  14    3:1     0       CURRENTBLOCK := CURRENTBLOCK + 1;
4384  14    3:1     8       IF CURRENTBLOCK > REFBLK THEN CURRENTBLOCK := 0;
4385  14    3:1    21       IF BLOCKREAD(REFFILE,REFLIST^,1,CURRENTBLOCK) <> 1 THEN;
4386  14    3:0    42     END (*GETNEXTBLOCK*) ;
4387  14    3:0    54
4388  14    2:0     0   BEGIN (*GETREFS*)
4389  14    2:1     0     IF (NREFS = 1) AND (REFBLK = 0) THEN EXIT(GETREFS);
4390  14    2:1    17     COUNT := 0;
4391  14    2:1    20     FOR BLOCKCOUNT := 0 TO REFBLK DO
4392  14    2:2    33       BEGIN
4393  14    2:3    33         IF CURRENTBLOCK < REFBLK THEN MAX := REFSPERBLK ELSE MAX := NREFS-1;
4394  14    2:3    56         FOR J := 1 TO MAX DO
4395  14    2:4    67           IF ID = REFLIST^[J].KEY THEN
4396  14    2:5    80             BEGIN GENWORD(REFLIST^[J].OFFSET); COUNT := COUNT + 1 END;
4397  14    2:3    04         IF BLOCKCOUNT < REFBLK THEN GETNEXTBLOCK;
4398  14    2:2    13       END;
4399  14    2:1    20     LIC := IC; IC := FIC; GENWORD(COUNT); IC := LIC;
4400  14    2:1    35     (*NOW FILL REST OF 8-WORD RECORD*)
4401  14    2:1    35     FOR J := 1 TO ((8 - (COUNT MOD 8)) MOD 8) DO GENWORD(0)
4402  14    2:0    53   END (* GETREFS *) ;
4403  14    2:0    82
4404  14    4:0     1   PROCEDURE GLOBALSEARCH(FCP: CTP);
4405  14    4:0     2     VAR NEEDEDBYLINKER: BOOLEAN;
4406  14    4:0     3
4407  14    4:0     0   BEGIN
4408  14    4:1     0     NEEDEDBYLINKER := TRUE;
4409  14    4:1     3     WITH LIREC,FCP^ DO
4410  14    4:2     6       CASE KLASS OF
4411  14    4:2    11         TYPES: NEEDEDBYLINKER := FALSE;
4412  14    4:2    16         KONST: IF (IDTYPE^.SIZE = 1) AND NOT INMODULE THEN
4413  14    4:4    27                  BEGIN LITYPE := CONSTDEF;
4414  14    4:5    31                    CONSTANT := VALUES.IVAL
4415  14    4:4    32                  END
4416  14    4:3    37                ELSE NEEDEDBYLINKER := FALSE;
4417  14    4:2    44         FORMALVARS,
```

```
4418  14    4:2    44
4419  14    4:3    44
4420  14    4:4    44
4421  14    4:5    48
4422  14    4:6    48
4423  14    4:7    53
4424  14    4:8    57
4425  14    4:7    57
4426  14    4:6    61
4427  14    4:7    63
4428  14    4:8    67
4429  14    4:9    74
4430  14    4:8    74
4431  14    4:9    80
4432  14    4:7    82
4433  14    4:6    86
4434  14    4:5    86
4435  14    4:4    90
4436  14    4:5    92
4437  14    4:6    96
4438  14    4:5    96
4439  14    4:3    02
4440  14    4:2    04
4441  14    4:2    09
4442  14    4:2    09
4443  14    4:4    09
4444  14    4:5    16
4445  14    4:6    23
4446  14    4:7    30
4447  14    4:8    35
4448  14    4:8    39
4449  14    4:7    45
4450  14    4:8    51
4451  14    4:9    55
4452  14    4:8    55
4453  14    4:6    61
4454  14    4:7    66
4455  14    4:8    71
4456  14    4:8    75
4457  14    4:7    81
4458  14    4:8    87
```

```
ACTUALVARS:
        BEGIN
          IF INMODULE THEN
            BEGIN
              IF PUBLIC THEN
                BEGIN LITYPE := PUBBLIC;
                  NWORDS := 0
                END
              ELSE
                BEGIN LITYPE := PRIVVATE;
                  IF KLASS = FORMALVARS THEN
                    NWORDS := PTRSIZE
                  ELSE
                    NWORDS := IDTYPE^.SIZE
                END;
              FORMAT := BIG
            END
          ELSE
            BEGIN LITYPE := PUBLICDEF;
              BASEOFFSET := VADDR
            END
        END;
FIELD: NEEDEDBYLINKER := FALSE;
PROC,
FUNC:  BEGIN
          IF PFDECKIND = DECLARED THEN
            IF PFKIND = ACTUAL THEN
              IF KLASS = PROC THEN
                IF EXTURNAL THEN
                  IF SEPPROC THEN LITYPE := SEPPREF
                  ELSE LITYPE := EXTPROC
                ELSE
                  IF SEPPROC THEN
                    LITYPE := SSEPPROC
                  ELSE NEEDEDBYLINKER := FALSE
              ELSE (*KLASS = FUNC*)
                IF EXTURNAL THEN
                  IF SEPPROC THEN LITYPE := SEPFREF
                  ELSE LITYPE := EXTFUNC
                ELSE
                  IF SEPPROC THEN
```

```
4459  14    4:9    91                          LITYPE := SSEPFUNC
4460  14    4:8    91                        ELSE NEEDEDBYLINKER := FALSE
4461  14    4:5    97                    ELSE NEEDEDBYLINKER := FALSE
4462  14    4:4    02                  ELSE NEEDEDBYLINKER := FALSE;
4463  14    4:4    10                  IF NEEDEDBYLINKER THEN
4464  14    4:5    13                    BEGIN
4465  14    4:6    13                      LCP := NEXT; NPARAMS := 0;
4466  14    4:6    22                      WHILE LCP <> NIL DO
4467  14    4:7    29                        BEGIN
4468  14    4:8    29                          WITH LCP^ DO
4469  14    4:9    34                            IF KLASS = FORMALVARS THEN
4470  14    4:0    41                              NPARAMS := NPARAMS + PTRSIZE
4471  14    4:9    44                            ELSE
4472  14    4:0    51                              IF KLASS = ACTUALVARS THEN
4473  14    4:1    58                                IF IDTYPE^.FORM <= POWER THEN
4474  14    4:2    65                                  NPARAMS := NPARAMS + IDTYPE^.SIZE
4475  14    4:1    70                                ELSE NPARAMS := NPARAMS + PTRSIZE;
4476  14    4:8    85                          LCP := LCP^.NEXT
4477  14    4:7    88                        END;
4478  14    4:6    94                      IF LITYPE IN [SEPPREF,SEPFREF] THEN
4479  14    4:7    04                        BEGIN FORMAT := BYTE; NWORDS := NPARAMS END
4480  14    4:6    14                      ELSE
4481  14    4:7    16                        BEGIN PROCNUM := PFNAME; RANGE := NIL END
4482  14    4:5    26                    END
4483  14    4:3    26                  END (*PROC,FUNC*);
4484  14    4:2    28        MODULE:   BEGIN
4485  14    4:4    28                    IF NOT INMODULE THEN NEEDEDBYLINKER := FALSE
4486  14    4:4    33                    ELSE
4487  14    4:5    38                      BEGIN LITYPE := MODDULE; NWORDS := 0; FORMAT := BYTE END
4488  14    4:3    50                  END
4489  14    4:2    50      END (*CASE,WITH*);
4490  14    4:1    76    IF NEEDEDBYLINKER THEN
4491  14    4:2    79      IF SEGTABLE[SEG].SEGKIND = 2 (*SEGPROC*) THEN
4492  14    4:3    90        WITH LIREC DO
4493  14    4:4    90          IF (LITYPE = CONSTDEF) OR (LITYPE = PUBLICDEF) THEN
4494  14    4:5    03            NEEDEDBYLINKER := FALSE;
4495  14    4:1    06    IF NEEDEDBYLINKER THEN
4496  14    4:2    09      WITH LIREC DO
4497  14    4:3    09        BEGIN LINAME := FCP^.NAME;
4498  14    4:4    15          FOR LGTH := 1 TO 8 DO GENBYTE(ORD(LINAME[LGTH]));
4499  14    4:4    46          GENWORD(ORD(LITYPE));
```

```
4500  14    4:4    52          CASE LITYPE OF
4501  14    4:4    57            MODDULE,
4502  14    4:4    57            PUBBLIC,
4503  14    4:4    57            PRIVVATE,
4504  14    4:4    57            SEPPREF,SEPFREF: BEGIN
4505  14    4:6    57                               GENWORD(ORD(FORMAT));
4506  14    4:6    63                               FIC := IC; GENWORD(0);
4507  14    4:6    71                               GENWORD(NWORDS);
4508  14    4:6    77                               IF LITYPE = MODDULE THEN GETREFS(FCP^.SEGID,1)
4509  14    4:6    88                               ELSE
4510  14    4:7    92                                IF LITYPE IN [SEPPREF,SEPFREF] THEN
4511  14    4:8    02                                  GETREFS(-FCP^.PFNAME,1)
4512  14    4:7    07                                ELSE GETREFS(FCP^.VADDR + 32,FCP^.IDTYPE^.SIZE);
4513  14    4:5    21                             END;
4514  14    4:4    23            CONSTDEF: BEGIN  GENWORD(CONSTANT); GENWORD(0); GENWORD(0) END;
4515  14    4:4    39            PUBLICDEF: BEGIN GENWORD(BASEOFFSET); GENWORD(0); GENWORD(0) END;
4516  14    4:4    55            EXTPROC,EXTFUNC:     BEGIN
4517  14    4:6    55                                   GENWORD(PROCNUM);
4518  14    4:6    61                                   GENWORD(NPARAMS);
4519  14    4:6    67                                   GENWORD(ORD(RANGE))
4520  14    4:5    70                                 END;
4521  14    4:4    75            SSEPPROC,SSEPFUNC:   BEGIN
4522  14    4:6    75                                   GENWORD(PROCNUM);
4523  14    4:6    81                                   GENWORD(NPARAMS);
4524  14    4:6    87                                   GENWORD(ORD(RANGE));
4525  14    4:6    93                                   FOR LGTH := 1 TO 8 DO
4526  14    4:7    05                                     GENBYTE(ORD(LINAME[LGTH]));
4527  14    4:6    24                                   IF LITYPE = SSEPPROC THEN
4528  14    4:7    31                                     GENWORD(ORD(SEPPREF))
4529  14    4:6    32                                   ELSE GENWORD(ORD(SEPFREF));
4530  14    4:6    41                                   GENWORD(ORD(BYTE));
4531  14    4:6    45                                   FIC := IC; GENWORD(0); GENWORD(NPARAMS);
4532  14    4:6    59                                   GETREFS(-PROCNUM,1)
4533  14    4:5    64                                 END
4534  14    4:4    66          END(*CASE*)
4535  14    4:3    04        END(*WITH*);
4536  14    4:1    04    IF IC >= 1024 THEN BEGIN WRITECODE(FALSE); IC := 0 END;
4537  14    4:1    18
4538  14    4:1    18    IF FCP^.LLINK <> NIL THEN GLOBALSEARCH(FCP^.LLINK);
4539  14    4:1    28    IF FCP^.RLINK <> NIL THEN GLOBALSEARCH(FCP^.RLINK)
4540  14    4:1    36
```

```
4541  14    4:0    36   END (*GLOBALSEARCH*);
4542  14    4:0    66
4543  14    1:0     0 BEGIN (*WRITELINKERINFO*)
4544  14    1:1     0   IC := 0;
4545  14    1:1     3   IF CODEINSEG THEN ERROR(399);
4546  14    1:1    13   IF INMODULE THEN
4547  14    1:2    17       CURRENTBLOCK := REFBLK;
4548  14    1:1    22   IF DECSTUFF THEN (*SKIP IF NO DECLARATIONPART LINKER INFO*)
4549  14    1:2    25     BEGIN FCP := DISPLAY[GLEV].FNAME;
4550  14    1:3    34       IF FCP <> NIL THEN GLOBALSEARCH(FCP)
4551  14    1:2    40     END;
4552  14    1:2    42   (*NOW DO NONRESIDENT PROCS*)
4553  14    1:1    42   WITH LIREC DO
4554  14    1:2    42     FOR I := SEEK TO DECOPS DO
4555  14    1:3    54       IF PFNUMOF[I] <> 0 THEN
4556  14    1:4    65         BEGIN
4557  14    1:5    65           CASE I OF
4558  14    1:5    68             SEEK:       BEGIN LINAME := 'FSEEK   '; NPARAMS := 2 END;
4559  14    1:5    88             FREADREAL:  BEGIN LINAME := 'FREADREA'; NPARAMS := 2 END;
4560  14    1:5    08             FWRITEREAL: BEGIN LINAME := 'FWRITERE'; NPARAMS := 5 END;
4561  14    1:5    28             FREADDEC:   BEGIN LINAME := 'FREADDEC'; NPARAMS := 3 END;
4562  14    1:5    48             FWRITEDEC:  BEGIN LINAME := 'FWRITEDE';
4563  14    1:7    63                                    NPARAMS := 2+DECSIZE(MAXDEC) END;
4564  14    1:5    75             DECOPS:     BEGIN LINAME := 'DECOPS  '; NPARAMS := 0 END;
4565  14    1:5    95           END;
4566  14    1:5    14           FOR LGTH := 1 TO 8 DO GENBYTE(ORD(LINAME[LGTH]));
4567  14    1:5    45           IF SEPPROC THEN
4568  14    1:6    49             BEGIN GENWORD(ORD(SEPPREF));
4569  14    1:7    53               GENWORD(ORD(BYTE)); FIC := IC; GENWORD(0); GENWORD(NPARAMS);
4570  14    1:7    69               GETREFS(-PFNUMOF[I],1)
4571  14    1:6    78             END
4572  14    1:5    80           ELSE
4573  14    1:6    82             BEGIN GENWORD(ORD(EXTPROC));
4574  14    1:7    86               GENWORD(PFNUMOF[I]); GENWORD(NPARAMS); GENWORD(0)
4575  14    1:6    02             END;
4576  14    1:5    05           PFNUMOF[I] := 0;
4577  14    1:4    13         END;
4578  14    1:4    20   (* NOW DO EOFMARK END-RECORD*)
4579  14    1:1    20   FOR LGTH := 1 TO 8 DO GENBYTE(ORD(' '));
4580  14    1:1    45   GENWORD(ORD(EOFMARK)); GENWORD(LCMAX);
4581  14    1:1    54   GENWORD(0);GENWORD(0);
```

```
4582  14    1:1    62    WRITECODE(TRUE);
4583  14    1:1    66    CLINKERINFO := FALSE;
4584  14    1:1    69    IF DECSTUFF THEN CLINKERINFO := FALSE
4585  14    1:0    72  END (*WRITELINKERINFO*);
4586  14    1:0    98
4587  15    1:0     1  SEGMENT PROCEDURE UNITPART(FSYS: SETOFSYS);
4588  15    1:0     5    VAR UMARKP: TESTP;
4589  15    1:0     6
4590  15    2:0     1    PROCEDURE OPENREFFILE;
4591  15    2:0     0    BEGIN
4592  15    2:1     0      REWRITE(REFFILE,'*SYSTEM.INFO[*]');
4593  15    2:1    26      IF IORESULT <> 0 THEN ERROR(402)
4594  15    2:0    35    END (* OPENREFFILE *) ;
4595  15    2:0    50
4596  15    3:0     1    PROCEDURE UNITDECLARATION(FSYS: SETOFSYS; VAR UMARKP:TESTP);
4597  15    3:0     6      VAR LCP: CTP; FOUND: BOOLEAN; LLEXSTK: LEXSTKREC;
4598  15    3:0     0    BEGIN
4599  15    3:1     0      IF INMODULE THEN ERROR(182 (* NESTED MODULES NOT ALLOWED *));
4600  15    3:1    10      IF CODEINSEG THEN
4601  15    3:2    14        BEGIN ERROR(399); SEGINX := 0; CURBYTE := 0 END;
4602  15    3:1    27      WITH LLEXSTK DO
4603  15    3:2    27        BEGIN
4604  15    3:3    27          DOLDTOP := TOP;
4605  15    3:3    30          DOLDLEV := LEVEL;
4606  15    3:3    34          POLDPROC := CURPROC;
4607  15    3:3    38          SOLDPROC := NEXTPROC;
4608  15    3:3    42          DOLDSEG := SEG;
4609  15    3:3    45          DLLC := LC;
4610  15    3:3    48          PREVLEXSTACKP := TOS
4611  15    3:2    48        END;
4612  15    3:1    52      SEG := NEXTSEG;
4613  15    3:1    56      NEXTSEG := NEXTSEG + 1;
4614  15    3:1    62      IF NEXTSEG > MAXSEG THEN ERROR(250);
4615  15    3:1    74      NEXTPROC := 1;
4616  15    3:1    77      LC := LCAFTERMARKSTACK;
4617  15    3:1    80      PUBLICPROCS := FALSE;
4618  15    3:1    83      INMODULE := TRUE;
4619  15    3:1    86      INSYMBOL;
4620  15    3:1    89      IF SY <> IDENT THEN ERROR(2)
4621  15    3:1    95      ELSE
4622  15    3:2    00        BEGIN FOUND := FALSE;
```

```
4623  15    3:3    03          LCP := MODPTR;
4624  15    3:3    07          WHILE (LCP <> NIL) AND NOT FOUND DO
4625  15    3:4    15            IF LCP^.NAME <> ID THEN LCP := LCP^.NEXT
4626  15    3:4    24            ELSE BEGIN FOUND := TRUE; ERROR(101) END;
4627  15    3:3    38          IF NOT FOUND THEN
4628  15    3:4    42            BEGIN NEW(LCP,MODULE);
4629  15    3:5    47              WITH LCP^ DO
4630  15    3:6    50                BEGIN NAME := ID; IDTYPE := NIL; NEXT := MODPTR;
4631  15    3:7    69                  KLASS := MODULE; SEGID := SEG
4632  15    3:6    79                END;
4633  15    3:5    81              MODPTR := LCP
4634  15    3:4    81            END;
4635  15    3:2    84        END;
4636  15    3:1    84      SEGTABLE[SEG].SEGNAME := ID;
4637  15    3:1    96      MARK(UMARKP);
4638  15    3:1    99      NEW(REFLIST);
4639  15    3:1    07      NEW(TOS);
4640  15    3:1    12      TOS^ := LLEXSTK;
4641  15    3:1    18      LEVEL := 1;
4642  15    3:1    21      IF TOP < DISPLIMIT THEN
4643  15    3:2    26        BEGIN TOP := TOP +1;
4644  15    3:3    31          WITH DISPLAY[TOP] DO
4645  15    3:4    38            BEGIN FNAME := NIL; FFILE := NIL; FLABEL := NIL; OCCUR := BLCK END;
4646  15    3:3    60          IF LCP <> NIL THEN ENTERID(LCP)
4647  15    3:2    66        END
4648  15    3:1    69      ELSE ERROR(250);
4649  15    3:1    77      INSYMBOL;
4650  15    3:1    80      IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
4651  15    3:0    91    END (*UNITDECLARATION*) ;
4652  15    3:0    08
4653  15    1:0     0  BEGIN (*UNITPART*)
4654  15    1:1     0    OPENREFFILE;
4655  15    1:1     2    REPEAT
4656  15    1:2     2      RESET(REFFILE); NREFS := 1; REFBLK := 0;
4657  15    1:2    16      IF (SY = SEPARATSY) THEN
4658  15    1:3    21        BEGIN SEPPROC := TRUE;
4659  15    1:4    24          INSYMBOL; IF SY <> UNITSY THEN ERROR(24)
4660  15    1:3    33        END
4661  15    1:2    36      ELSE
4662  15    1:3    38        SEPPROC := FALSE;
4663  15    1:2    41      UNITDECLARATION(FSYS,UMARKP);
```

```
4664  15    1:2    52     IF SEPPROC THEN SEGTABLE[SEG].SEGKIND := 4 ELSE SEGTABLE[SEG].SEGKIND := 3;
4665  15    1:2    78     SEGTABLE[SEG].TEXTADDR := CURBLK;
4666  15    1:2    90     WRITETEXT;
4667  15    1:2    93     IF SY = INTERSY THEN INSYMBOL
4668  15    1:2    98     ELSE ERROR(22);
4669  15    1:2    07     ININTERFACE := TRUE;
4670  15    1:2    10     DECLARATIONPART(FSYS);
4671  15    1:2    20     IF PUBLICPROCS THEN
4672  15    1:3    24       BEGIN
4673  15    1:4    24         ININTERFACE := FALSE;
4674  15    1:4    27         IF SY <> IMPLESY THEN BEGIN ERROR(23); SKIP(FSYS - STATBEGSYS) END
4675  15    1:4    52         ELSE INSYMBOL;
4676  15    1:4    57         BLOCK(FSYS - [SEPARATSY,UNITSY,INTERSY,IMPLESY]);
4677  15    1:4    79         IF REFBLK > 0 THEN
4678  15    1:5    86           IF BLOCKWRITE(REFFILE,REFLIST^,1,REFBLK) <> 1 THEN ERROR(402);
4679  15    1:4    13         WRITELINKERINFO(TRUE);
4680  15    1:3    17       END
4681  15    1:2    17     ELSE
4682  15    1:3    19       BEGIN DLINKERINFO := FALSE;
4683  15    1:4    22         WITH SEGTABLE[SEG] DO
4684  15    1:5    30           BEGIN CODELENG := 0; DISKADDR :=CURBLK; SEGKIND := 0 END;
4685  15    1:3    45       END;
4686  15    1:2    45     SEPPROC := FALSE; (*FALSE WHENEVER NOT INMODULE*)
4687  15    1:2    48     INMODULE := FALSE;
4688  15    1:2    51     IF SY = ENDSY THEN INSYMBOL
4689  15    1:2    56     ELSE BEGIN ERROR(13); SKIP(FSYS) END;
4690  15    1:2    75     IF SY <> PERIOD THEN
4691  15    1:3    80       IF SY = SEMICOLON THEN INSYMBOL
4692  15    1:3    85       ELSE ERROR(14);
4693  15    1:2    94     WITH TOS^ DO
4694  15    1:3    98         BEGIN
4695  15    1:4    98           TOP := DOLDTOP;
4696  15    1:4    02           LEVEL := DOLDLEV;
4697  15    1:4    06           CURPROC := POLDPROC;
4698  15    1:4    10           NEXTPROC := SOLDPROC;
4699  15    1:4    14           SEG := DOLDSEG;
4700  15    1:4    18           LC := DLLC;
4701  15    1:3    22         END;
4702  15    1:2    22     TOS := TOS^.PREVLEXSTACKP;
4703  15    1:2    28     RELEASE(UMARKP)
4704  15    1:1    30   UNTIL NOT (SY IN [UNITSY,SEPARATSY]);
```

```
4705  15    1:1    49   CLOSE(REFFILE)
4706  15    1:0    56 END (*UNITPART*);
4707  15    1:0    70
4708  15    1:0    70
4709  15    1:0    70 (*$I #5:UNITPART.TEXT*)
4709  15    1:0    70 (*$I #5:PROCS.A.TEXT*)
4710  15    1:0    70
4711  15    1:0    70     (*************************************************************)
4712  15    1:0    70     (*                                                           *)
4713  15    1:0    70     (*  COPYRIGHT (C) 1979 REGENTS OF THE UNIVERSITY OF CALIFORNIA.   *)
4714  15    1:0    70     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-    *)
4715  15    1:0    70     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE   *)
4716  15    1:0    70     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
4717  15    1:0    70     (*                                                           *)
4718  15    1:0    70     (*************************************************************)
4719  15    1:0    70
4720  10    2:D     1 PROCEDURE ERROR(*ERRORNUM: INTEGER*);
4721  10    2:D     2   VAR CH: CHAR; ERRSTART: INTEGER;
4722  10    2:D     4       A: PACKED ARRAY [0..179] OF CHAR;
4723  10    2:0     0 BEGIN
4724  10    2:1     0   WITH USERINFO DO
4725  10    2:2     3     IF (ERRSYM <> SYMCURSOR) OR (ERRBLK <> SYMBLK) THEN
4726  10    2:3    17       BEGIN ERRBLK := SYMBLK;
4727  10    2:4    24         ERRSYM := SYMCURSOR; ERRNUM := ERRORNUM;
4728  10    2:4    36         IF STUPID THEN CH := 'E'
4729  10    2:4    41         ELSE
4730  10    2:5    46           BEGIN
4731  10    2:6    46             IF NOISY THEN WRITELN(OUTPUT)
4732  10    2:6    56             ELSE
4733  10    2:7    58               IF LIST AND (ERRORNUM <= 400) THEN
4734  10    2:8    68                 EXIT(ERROR);
4735  10    2:6    72             IF LINESTART = 0 THEN
4736  10    2:7    78               WRITE(OUTPUT,SYMBUFP^:SYMCURSOR)
4737  10    2:6    69             ELSE
4738  10    2:7    91               BEGIN
4739  10    2:8    91                 ERRSTART := SCAN(-(LINESTART-1),=CHR(EOL),
4740  10    2:8    98                                  SYMBUFP^[LINESTART-2])+LINESTART-1;
4741  10    2:8    13                 MOVELEFT(SYMBUFP^[ERRSTART],A[0],SYMCURSOR-ERRSTART);
4742  10    2:8    23                 WRITE(OUTPUT,A:SYMCURSOR-ERRSTART)
4743  10    2:7    37               END;
4744  10    2:6    37             WRITELN(OUTPUT,' <<<<');
```

```
4745  10    2:6    58              WRITE(OUTPUT,'LINE ',SCREENDOTS,', ERROR ',ERRORNUM:0,':');
4746  10    2:6    16              IF NOISY THEN
4747  10    2:7    20                WRITE(OUTPUT,' <SP>(CONTINUE), <ESC>(TERMINATE), E(DIT');
4748  10    2:6    70              WRITE(OUTPUT,CHR(7));
4749  10    2:6    78              REPEAT READ(KEYBOARD,CH)
4750  10    2:6    86              UNTIL (CH = ' ') OR (CH = 'E') OR (CH = 'E') OR (CH = ALTMODE)
4751  10    2:5    02            END;
4752  10    2:4    05          IF (CH = 'E') OR (CH = 'E') THEN
4753  10    2:5    14            BEGIN ERRBLK := SYMBLK-2; EXIT(PASCALCOMPILER) END;
4754  10    2:4    27          IF (ERRORNUM > 400) OR (CH = CHR(27)) THEN
4755  10    2:5    38            BEGIN ERRBLK := 0; EXIT(PASCALCOMPILER) END;
4756  10    2:4    48          WRITELN(OUTPUT);
4757  10    2:4    54          IF NOISY THEN
4758  10    2:5    58            WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
4759  10    2:3    83        END
4760  10    2:0    83 END (*ERROR*) ;
4761  10    2:0    02
4762  10    3:D     1 PROCEDURE GETNEXTPAGE;
4763  10    3:0     0 BEGIN SYMCURSOR := 0; LINESTART := 0;
4764  10    3:1     6   IF USING THEN
4765  10    3:2    10     BEGIN
4766  10    3:3    10       IF USEFILE = WORKCODE THEN
4767  10    3:4    17         BEGIN
4768  10    3:5    17           IF BLOCKREAD(USERINFO.WORKCODE^,SYMBUFP^,2,SYMBLK) <> 2 THEN
4769  10    3:6    34             USING := FALSE
4770  10    3:4    34         END
4771  10    3:3    37       ELSE
4772  10    3:4    39         IF USEFILE = SYSLIBRARY THEN
4773  10    3:5    46           IF BLOCKREAD(LIBRARY,SYMBUFP^,2,SYMBLK) <> 2 THEN
4774  10    3:6    64             USING := FALSE;
4775  10    3:3    67       IF NOT USING THEN
4776  10    3:4    72         BEGIN
4777  10    3:5    72           SYMBLK := PREVSYMBLK; SYMCURSOR := PREVSYMCURSOR;
4778  10    3:5    82           LINESTART := PREVLINESTART
4779  10    3:4    82         END
4780  10    3:2    87     END;
4781  10    3:1    87   IF NOT USING THEN
4782  10    3:2    92     BEGIN
4783  10    3:3    92       IF INCLUDING THEN
4784  10    3:4    96         IF BLOCKREAD(INCLFILE,SYMBUFP^,2,SYMBLK) <> 2 THEN
4785  10    3:5    14           BEGIN CLOSE(INCLFILE); INCLUDING := FALSE;
```

```
4786  10    3:6    24              SYMBLK := OLDSYMBLK; SYMCURSOR := OLDSYMCURSOR;
4787  10    3:6    34              LINESTART := OLDLINESTART
4788  10    3:5    34            END
4789  10    3:2    39      END;
4790  10    3:1    39    IF NOT (INCLUDING OR USING) THEN
4791  10    3:2    47      IF BLOCKREAD(USERINFO.WORKSYM^,SYMBUFP^,2,SYMBLK) <> 2 THEN
4792  10    3:3    64        ERROR(401);
4793  10    3:1    69    IF SYMCURSOR = 0 THEN
4794  10    3:2    74      BEGIN
4795  10    3:3    74        IF INMODULE THEN
4796  10    3:4    78          IF ININTERFACE AND NOT USING THEN WRITETEXT;
4797  10    3:3    88        IF SYMBUFP^[0] = CHR(16(*DLE*)) THEN
4798  10    3:4    95         SYMCURSOR := 2
4799  10    3:2    95      END;
4800  10    3:1    98    SYMBLK := SYMBLK+2
4801  10    3:0    00  END (*GETNEXTPAGE*) ;
4802  10    3:0    16
4803  10    3:0    16  (*$I+*)
4804  10    4:D     1  PROCEDURE PRINTLINE;
4805  10    4:D     1    VAR DORLEV,STARORC: CHAR; LENG: INTEGER;
4806  10    4:D     4        A: PACKED ARRAY [0..99] OF CHAR;
4807  10    4:0     0  BEGIN STARORC := ':';
4808  10    4:1     3    IF DP THEN DORLEV := 'D'
4809  10    4:1     7    ELSE DORLEV := CHR((BEGSTMTLEV MOD 10) + ORD('0'));
4810  10    4:1    20    IF BPTONLINE THEN STARORC := '*';
4811  10    4:1    27    WRITE(LP,SCREENDOTS:6,SEG:4,CURPROC:5,
4812  10    4:1    59                STARORC,DORLEV,LINEINFO:6,' ');
4813  10    4:1    00    LENG := SYMCURSOR-LINESTART;
4814  10    4:1    06    IF LENG > 100 THEN LENG := 100;
4815  10    4:1    14    MOVELEFT(SYMBUFP^[LINESTART],A,LENG);
4816  10    4:1    23    IF A[0] = CHR(16(*DLE*)) THEN
4817  10    4:2    31      BEGIN
4818  10    4:3    31        IF A[1] > ' ' THEN
4819  10    4:4    39          WRITE(LP,' ':ORD(A[1])-ORD(' '));
4820  10    4:3    54        LENG := LENG-2;
4821  10    4:3    59        MOVELEFT(A[2],A,LENG)
4822  10    4:2    68      END;
4823  10    4:1    68    A[LENG-1] := CHR(EOL); (*JUST TO MAKE SURE*)
4824  10    4:1    75    WRITE(LP,A:LENG);
4825  10    4:1    87    WITH USERINFO DO
4826  10    4:2    90      IF (ERRBLK = SYMBLK) AND (ERRSYM > LINESTART) THEN
```

```
4827  10     4:3     05        WRITELN(LP,'>>>>>> ERROR # ',ERRNUM)
4828  10     4:0     52 END (*PRINTLINE*) ;
4829  10     4:0     64 (*$I-*)
4830  10     4:0     64
4831  10     5:0      1 PROCEDURE ENTERID(*FCP: CTP*);
4832  10     5:0      2   VAR LCP,LCP1: CTP; I: INTEGER;
4833  10     5:0      0 BEGIN LCP := DISPLAY[TOP].FNAME;
4834  10     5:1      8   IF LCP = NIL THEN DISPLAY[TOP].FNAME := FCP
4835  10     5:1     18   ELSE
4836  10     5:2     22     BEGIN I := TREESEARCH(LCP,LCP1,FCP^.NAME);
4837  10     5:3     30       WHILE I = 0 DO
4838  10     5:4     35         BEGIN ERROR(101);
4839  10     5:5     38           IF LCP1^.RLINK = NIL THEN I := 1
4840  10     5:5     44           ELSE I := TREESEARCH(LCP1^.RLINK,LCP1,FCP^.NAME)
4841  10     5:4     56         END;
4842  10     5:3     60       IF I = 1 THEN LCP1^.RLINK := FCP ELSE LCP1^.LLINK := FCP
4843  10     5:2     75     END;
4844  10     5:1     77   FCP^.LLINK := NIL; FCP^.RLINK := NIL
4845  10     5:0     85 END (*ENTERID*) ;
4846  10     5:0     02
4847  10     6:0      1 PROCEDURE INSYMBOL; (* COMPILER VERSION 3.4 06-NOV-76 *)
4848  10     6:0      1   LABEL 1;
4849  10     6:0      1   VAR LVP: CSP; X: INTEGER;
4850  10     6:0      3
4851  10    21:0      1 PROCEDURE CHECKEND;
4852  10    21:0      0 BEGIN (* CHECKS FOR THE END OF THE PAGE *)
4853  10    21:1      0   SCREENDOTS := SCREENDOTS+1;
4854  10    21:1      6   SYMCURSOR := SYMCURSOR + 1;
4855  10    21:1     11   IF NOISY THEN
4856  10    21:2     15     BEGIN WRITE(OUTPUT,'.');
4857  10    21:3     23       IF (SCREENDOTS-STARTDOTS) MOD 50 = 0 THEN
4858  10    21:4     34         BEGIN WRITELN(OUTPUT);
4859  10    21:5     40           WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
4860  10    21:4     65         END
4861  10    21:2     65     END;
4862  10    21:1     65   IF LIST THEN PRINTLINE;
4863  10    21:1     71   BPTONLINE := FALSE;
4864  10    21:1     74   IF SYMBUFP^[SYMCURSOR]=CHR(0) THEN GETNEXTPAGE
4865  10    21:1     81   ELSE LINESTART := SYMCURSOR;
4866  10    21:1     88   IF SYMBUFP^[SYMCURSOR] = CHR(12(*FF*)) THEN SYMCURSOR:=SYMCURSOR+1;
4867  10    21:1     00   IF SYMBUFP^[SYMCURSOR] = CHR(16(*DLE*)) THEN
```

```
4868  10   21:2   07     SYMCURSOR := SYMCURSOR+2
4869  10   21:1   08   ELSE
4870  10   21:2   14     BEGIN
4871  10   21:3   14       SYMCURSOR := SYMCURSOR+SCAN(80,<>CHR(9),SYMBUFP^[SYMCURSOR]);
4872  10   21:3   26       SYMCURSOR := SYMCURSOR+SCAN(80,<>' ',SYMBUFP^[SYMCURSOR])
4873  10   21:2   55     END;
4874  10   21:1   38   IF DP THEN LINEINFO := LC ELSE LINEINFO := IC
4875  10   21:0   47 END;
4876  10   21:0   62
4877  10   22:0    1 PROCEDURE COMMENTER(STOPPER: CHAR);
4878  10   22:0    2   VAR CH,SW,DEL: CHAR; LTITLE: STRING[40];
4879  10   22:0   26
4880  10   23:0    1   PROCEDURE SCANSTRING(VAR STRG: STRING; MAXLENG: INTEGER);
4881  10   23:0    3     VAR LENG: INTEGER;
4882  10   23:0    0   BEGIN SYMCURSOR := SYMCURSOR+2;
4883  10   23:1    5     LENG := SCAN(MAXLENG,=STOPPER,SYMBUFP^[SYMCURSOR]);
4884  10   23:1   17     STRG[0] := CHR(LENG);
4885  10   23:1   21     MOVELEFT(SYMBUFP^[SYMCURSOR],STRG[1],LENG);
4886  10   23:1   28     SYMCURSOR := SYMCURSOR+LENG+1
4887  10   23:0   31   END (*SCANSTRING*) ;
4888  10   23:0   48
4889  10   22:0    0 BEGIN
4890  10   22:1    0   SYMCURSOR := SYMCURSOR+1; (* POINT TO THE FIRST CH PAST "(*" *)
4891  10   22:1    5   IF SYMBUFP^[SYMCURSOR]='$' THEN
4892  10   22:2   12      IF SYMBUFP^[SYMCURSOR+1] <> STOPPER THEN
4893  10   22:3   21         REPEAT
4894  10   22:4   21           CH := SYMBUFP^[SYMCURSOR+1];
4895  10   22:4   28           SW := SYMBUFP^[SYMCURSOR+2];
4896  10   22:4   35           DEL := SYMBUFP^[SYMCURSOR+3];
4897  10   22:4   42           IF (SW = ',') OR (SW = STOPPER) THEN
4898  10   22:5   51             BEGIN DEL := SW; SW := '+';
4899  10   22:6   57               SYMCURSOR := SYMCURSOR-1
4900  10   22:5   58             END;
4901  10   22:4   62           CASE CH OF
4902  10   22:4   65           'C': BEGIN
4903  10   22:6   65                  IF LEVEL > 1 THEN ERROR(194);
4904  10   22:6   76                  NEW(COMMENT); SCANSTRING(COMMENT^,80); EXIT(COMMENTER)
4905  10   22:5   92                END;
4906  10   22:4   94           'D': DEBUGGING := (SW='+');
4907  10   22:4   01           'F': FLIPBYTES := (SW='+');
4908  10   22:4   08           'G': GOTOOK := (SW='+');
```

```
4909  10    22:4     15          'I': IF (SW='+') OR (SW='-') THEN IOCHECK := (SW='+')
4910  10    22:5     27               ELSE
4911  10    22:6     31                 BEGIN SCANSTRING(LTITLE,40);
4912  10    22:7     36                   IF STOPPER = '*' THEN
4913  10    22:8     41                     SYMCURSOR := SYMCURSOR+1;
4914  10    22:7     46                   IF LIST THEN
4915  10    22:8     50                     BEGIN
4916  10    22:9     50                       SYMCURSOR := SYMCURSOR + 1;
4917  10    22:9     55                       PRINTLINE;
4918  10    22:9     57                       SYMCURSOR := SYMCURSOR - 1;
4919  10    22:8     62                     END;
4920  10    22:7     62                   IF INCLUDING OR INMODULE AND ININTERFACE THEN
4921  10    22:8     72                     BEGIN ERROR(406); EXIT(COMMENTER) END;
4922  10    22:7     81                   OPENOLD(INCLFILE,LTITLE);
4923  10    22:7     91                   IF IORESULT <> 0 THEN
4924  10    22:8     97                     BEGIN OPENOLD(INCLFILE,CONCAT(LTITLE,'.TEXT'));
4925  10    22:9     32                        IF IORESULT <> 0 THEN ERROR(403)
4926  10    22:8     41                     END;
4927  10    22:7     43                   INCLUDING := TRUE;
4928  10    22:7     46                   OLDSYMCURSOR := SYMCURSOR;
4929  10    22:7     50                   OLDLINESTART := LINESTART;
4930  10    22:7     55                   OLDSYMBLK := SYMBLK-2;
4931  10    22:7     62                   SYMBLK := 2; GETNEXTPAGE;
4932  10    22:7     67                   INSYMBOL; EXIT(INSYMBOL)
4933  10    22:6     73                 END;
4934  10    22:4     75          'L': IF (SW='+') OR (SW='-') THEN
4935  10    22:6     84                 BEGIN LIST := (SW='+');
4936  10    22:7     89                   IF LIST THEN OPENNEW(LP,'*SYSTEM.LST.TEXT')
4937  10    22:6     20                 END
4938  10    22:5     20               ELSE
4939  10    22:6     22                 BEGIN SCANSTRING(LTITLE,40);
4940  10    22:7     27                   OPENNEW(LP,LTITLE);
4941  10    22:7     37                   LIST := IORESULT = 0;
4942  10    22:7     43                   EXIT(COMMENTER)
4943  10    22:6     47                 END;
4944  10    22:4     49          'Q': NOISY := (SW='-');
4945  10    22:4     56          'P': WRITE(LP,CHR(12(*FF*)));
4946  10    22:4     66          'R': RANGECHECK := (SW='+');
4947  10    22:4     73          'S': NOSWAP:=(SW='-');
4948  10    22:4     80          'T': TINY := (SW='+');
4949  10    22:4     87          'U': IF (SW='+') OR (SW='-') THEN
```

```
4950  10   22:6    96                    BEGIN SYSCOMP := (SW = '-');
4951  10   22:7    01                      RANGECHECK := NOT SYSCOMP;
4952  10   22:7    06                      IOCHECK := RANGECHECK;
4953  10   22:7    10                      GOTOOK := SYSCOMP
4954  10   22:6    10                    END
4955  10   22:5    14                  ELSE
4956  10   22:6    16                    IF NOT USING THEN
4957  10   22:7    21                      BEGIN SCANSTRING(SYSTEMLIB,40);
4958  10   22:8    27                        CLOSE(LIBRARY); LIBNOTOPEN := TRUE;
4959  10   22:8    37                        EXIT(COMMENTER)
4960  10   22:7    41                      END
4961  10   22:4    41              END (*CASES*);
4962  10   22:4    88              SYMCURSOR := SYMCURSOR+3;
4963  10   22:3    93            UNTIL DEL <> ',';
4964  10   22:1    98    SYMCURSOR := SYMCURSOR-1; (* ADJUST *)
4965  10   22:1    03    REPEAT
4966  10   22:2    03      REPEAT
4967  10   22:3    03        SYMCURSOR := SYMCURSOR+1;
4968  10   22:3    08        WHILE SYMBUFP^[SYMCURSOR] = CHR(EOL) DO CHECKEND
4969  10   22:2    15      UNTIL SYMBUFP^[SYMCURSOR]=STOPPER;
4970  10   22:1    26    UNTIL (SYMBUFP^[SYMCURSOR+1]=')') OR (STOPPER=']');
4971  10   22:1    39    SYMCURSOR := SYMCURSOR+1;
4972  10   22:0    44  END (*COMMENTER*);
4973  10   22:0    74
4974  10   24:D     1  PROCEDURE STRING;
4975  10   24:D     1  LABEL 1;
4976  10   24:D     1  VAR
4977  10   24:D     1    T: PACKED ARRAY [1..80] OF CHAR;
4978  10   24:D    41    TP,NBLANKS,L: INTEGER;
4979  10   24:D    44    DUPLE: BOOLEAN;
4980  10   24:D    45
4981  10   24:0     0  BEGIN
4982  10   24:1     0    DUPLE := FALSE; (* INDICATES WHEN '' IS PRESENT *)
4983  10   24:1     3    TP := 0; (* INDEX INTO TEMPORARY STRING *)
4984  10   24:1     6    REPEAT
4985  10   24:2     6      IF DUPLE THEN SYMCURSOR := SYMCURSOR+1;
4986  10   24:2    15      REPEAT
4987  10   24:3    15        SYMCURSOR := SYMCURSOR+1;
4988  10   24:3    20        TP := TP+1;
4989  10   24:3    26        IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
4990  10   24:4    33          BEGIN ERROR(202); CHECKEND; GOTO 1 END;
```

```
4991  10   24:3    42        T[TP] := SYMBUFP^[SYMCURSOR];
4992  10   24:2    52      UNTIL SYMBUFP^[SYMCURSOR]='''';
4993  10   24:2    59      DUPLE := TRUE;
4994  10   24:1    62    UNTIL SYMBUFP^[SYMCURSOR+1]<>'''';
4995  10   24:1    71 1:   TP := TP-1; (* ADJUST *)
4996  10   24:1    77    SY := STRINGCONST; OP := NOOP;
4997  10   24:1    83    LGTH := TP; (* GROSS *)
4998  10   24:1    87    IF TP=1 (* SINGLE CHARACTER CONSTANT *)
4999  10   24:1    89      THEN
5000  10   24:2    93        VAL.IVAL := ORD(T[1])
5001  10   24:1    99      ELSE
5002  10   24:2    03        WITH SCONST^ DO
5003  10   24:3    07          BEGIN
5004  10   24:4    07            CCLASS := STRG;
5005  10   24:4    11            SLGTH := TP;
5006  10   24:4    18            MOVELEFT(T[1],SVAL[1],TP);
5007  10   24:4    34            VAL.VALP := SCONST
5008  10   24:3    34          END
5009  10   24:0    38 END(*STRING*);
5010  10   24:0    54
5011  10   25:D     1 PROCEDURE NUMBER;
5012  10   25:D     1 VAR
5013  10   25:D     1   EXPONENT,ENDI,ENDF,ENDE,SIGN,IPART,FPART,EPART,
5014  10   25:D     1   ISUM:  INTEGER;
5015  10   25:D    10   TIPE: (REALTIPE,INTEGERTIPE);
5016  10   25:D    11   RSUM: REAL;
5017  10   25:D    13   NOTLONG: BOOLEAN;
5018  10   25:D    14   K,J: INTEGER;
5019  10   25:0     0 BEGIN
5020  10   25:0     0   (* TAKES A NUMBER AND DECIDES WHETHER IT'S REAL
5021  10   25:0     0      OR INTEGER AND CONVERTS IT TO THE INTERNAL
5022  10   25:0     0      FORM. *)
5023  10   25:1     0   TIPE := INTEGERTIPE;
5024  10   25:1     3   ENDI := 0;
5025  10   25:1     6   ENDF := 0;
5026  10   25:1     9   ENDE := 0;
5027  10   25:1    12   SIGN := 1;
5028  10   25:1    15   NOTLONG := TRUE;
5029  10   25:1    19   EPART := 9999; (* OUT OF REACH *)
5030  10   25:1    23   IPART := SYMCURSOR; (* INTEGER PART STARTS HERE *)
5031  10   25:1    26   REPEAT
```

```
5032  10   25:2   26      SYMCURSOR := SYMCURSOR+1
5033  10   25:1   27    UNTIL (SYMBUFP^[SYMCURSOR]<'0') OR (SYMBUFP^[SYMCURSOR]>'9');
5034  10   25:1   44    (* SYMCURSOR NOW POINTS AT FIRST CHARACTER PAST INTEGER PART *)
5035  10   25:1   44    ENDI := SYMCURSOR-1; (* MARK THE END OF IPART *)
5036  10   25:1   49    IF SYMBUFP^[SYMCURSOR]='.'
5037  10   25:1   52      THEN
5038  10   25:2   56        IF SYMBUFP^[SYMCURSOR+1]<>'.'  (* WATCH OUT FOR '..' *)
5039  10   25:2   61          THEN
5040  10   25:3   65            BEGIN
5041  10   25:4   65              TIPE := REALTIPE;
5042  10   25:4   68              SYMCURSOR := SYMCURSOR+1;
5043  10   25:4   73              FPART := SYMCURSOR; (* BEGINNING OF FPART *)
5044  10   25:4   76              WHILE (SYMBUFP^[SYMCURSOR] >= '0') AND
5045  10   25:4   81                    (SYMBUFP^[SYMCURSOR] <= '9') DO
5046  10   25:5   89                SYMCURSOR := SYMCURSOR+1;
5047  10   25:4   96              IF SYMCURSOR = FPART THEN ERROR(201);
5048  10   25:4   06              ENDF := SYMCURSOR-1;
5049  10   25:3   11            END;
5050  10   25:1   11    IF SYMBUFP^[SYMCURSOR]='E'
5051  10   25:1   14      THEN
5052  10   25:2   18        BEGIN
5053  10   25:3   18          TIPE := REALTIPE;
5054  10   25:3   21          SYMCURSOR := SYMCURSOR+1;
5055  10   25:3   26          IF SYMBUFP^[SYMCURSOR]='-'
5056  10   25:3   29            THEN
5057  10   25:4   33              BEGIN
5058  10   25:5   33                SYMCURSOR := SYMCURSOR+1;
5059  10   25:5   38                SIGN := -1;
5060  10   25:4   42              END
5061  10   25:3   42            ELSE
5062  10   25:4   44              IF SYMBUFP^[SYMCURSOR]='+'
5063  10   25:4   47                THEN
5064  10   25:5   51                  SYMCURSOR := SYMCURSOR+1;
5065  10   25:3   56          EPART := SYMCURSOR; (* BEGINNING OF EXPONENT *)
5066  10   25:3   59          WHILE (SYMBUFP^[SYMCURSOR]>='0') AND (SYMBUFP^[SYMCURSOR]<='9') DO
5067  10   25:4   72            SYMCURSOR := SYMCURSOR+1;
5068  10   25:3   79          ENDE := SYMCURSOR-1;
5069  10   25:3   84          IF ENDE<EPART THEN ERROR(201); (* ERROR IN REAL CONSTANT *)
5070  10   25:2   94        END;
5071  10   25:2   94    (* NOW CONVERT TO INTERNAL FORM *)
5072  10   25:1   94    IF TIPE=INTEGERTIPE THEN
```

```
5073  10   25:2   99      BEGIN
5074  10   25:3   99        ISUM := 0;
5075  10   25:3   02        FOR J := IPART TO ENDI DO
5076  10   25:4   13          BEGIN
5077  10   25:5   13            IF (ISUM>MAXINT DIV 10) OR ((ISUM=MAXINT DIV 10) AND
5078  10   25:5   27                          (ORD(SYMBUFP^[J]) - ORD('0') > MAXINT MOD 10)) THEN
5079  10   25:6   42              BEGIN NOTLONG := FALSE; K := J; J := ENDI END
5080  10   25:5   51            ELSE ISUM := ISUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
5081  10   25:4   64          END;
5082  10   25:3   71          IF NOTLONG THEN
5083  10   25:4   74            BEGIN
5084  10   25:5   74              SY := INTCONST;  OP := NOOP;
5085  10   25:5   80              VAL.IVAL := ISUM;
5086  10   25:4   83            END
5087  10   25:3   83          ELSE
5088  10   25:4   85            BEGIN
5089  10   25:5   85              IF ENDI - IPART >= MAXDEC THEN
5090  10   25:6   92                BEGIN ERROR(203); IPART := ENDI; K := ENDI END;
5091  10   25:5   03              NEW(LVP,LONG);
5092  10   25:5   09              WITH LVP^ DO
5093  10   25:6   14                BEGIN CCLASS := LONG; J := 4; LLENG := 0;
5094  10   25:7   25                  WHILE K <= ENDI DO
5095  10   25:8   30                    BEGIN
5096  10   25:9   30                      IF J = 4 THEN
5097  10   25:0   35                        BEGIN LLENG := LLENG + 1;
5098  10   25:1   43                          LONGVAL[LLENG] := ISUM;
5099  10   25:1   54                          ISUM := 0;
5100  10   25:1   57                          J := 0
5101  10   25:0   57                        END;
5102  10   25:9   60                      ISUM := ISUM * 10 + ORD(SYMBUFP^[K])-ORD('0');
5103  10   25:9   71                      K := K + 1; J := J + 1
5104  10   25:8   77                    END;
5105  10   25:7   83                  LLAST := J;
5106  10   25:7   88                  IF J > 0 THEN
5107  10   25:8   93                    BEGIN LLENG := LLENG + 1;
5108  10   25:9   01                      LONGVAL[LLENG] := ISUM
5109  10   25:8   10                    END;
5110  10   25:6   12                END;
5111  10   25:5   12              SY := LONGCONST; OP := NOOP;
5112  10   25:5   18              LGTH := ENDI - IPART + 1;
5113  10   25:5   25              VAL.VALP := LVP
```

```
5114  10   25:4   25            END;
5115  10   25:2   30      END (*TIPE = INTEGERTIPE*)
5116  10   25:1   30    ELSE
5117  10   25:2   32      BEGIN (* REAL NUMBER HERE *)
5118  10   25:3   32        RSUM := 0;
5119  10   25:3   38        FOR J := IPART TO ENDI DO
5120  10   25:4   49          BEGIN
5121  10   25:5   49            RSUM := RSUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
5122  10   25:4   67          END;
5123  10   25:3   74        FOR J := ENDF DOWNTO FPART DO
5124  10   25:4   85          RSUM := RSUM+(ORD(SYMBUFP^[J])-ORD('0'))/PWROFTEN(J-FPART+1);
5125  10   25:3   15        EXPONENT := 0;
5126  10   25:3   18        FOR J := EPART TO ENDE DO
5127  10   25:4   29          EXPONENT := EXPONENT*10+ORD(SYMBUFP^[J])-ORD('0');
5128  10   25:3   47        IF SIGN=-1 THEN
5129  10   25:4   53          RSUM := RSUM/PWROFTEN(EXPONENT)
5130  10   25:3   60        ELSE
5131  10   25:4   67          RSUM := RSUM*PWROFTEN(EXPONENT);
5132  10   25:3   79        SY := REALCONST;  OP := NOOP;
5133  10   25:3   85        NEW(LVP,REEL);
5134  10   25:3   91        LVP^.CCLASS := REEL;
5135  10   25:3   96        LVP^.RVAL := RSUM;
5136  10   25:3   07        VAL.VALP := LVP;
5137  10   25:2   12      END;
5138  10   25:1   12    SYMCURSOR := SYMCURSOR-1; (* ADJUST FOR POSTERITY *)
5139  10   25:0   17  END (*NUMBER*) ;
5140  10   25:0   52
5141  10    6:0    0  BEGIN (* INSYMBOL *)
5142  10    6:1    0    IF GETSTMTLEV THEN BEGIN BEGSTMTLEV := STMTLEV; GETSTMTLEV := FALSE END;
5143  10    6:1   11    OP := NOOP;
5144  10    6:1   14  1:  SY := OTHERSY; (* IF NO CASES EXERCISED BLOW UP *)
5145  10    6:1   17    CASE SYMBUFP^[SYMCURSOR] OF
5146  10    6:1   22    '''':STRING;
5147  10    6:1   26    '0','1','2','3','4','5','6','7','8','9':
5148  10    6:2   26         NUMBER;
5149  10    6:1   30    'A','B','C','D','E','F','G','H','I','J','K','L','M',
5150  10    6:1   30    'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
5151  10    6:1   30    'A','B','C','D','E','F','G','H','I','J','K','L','M',
5152  10    6:1   30    'N','O','P','Q','R','S','T','U','V','W','X','Y','Z':
5153  10    6:2   30         IDSEARCH(SYMCURSOR,SYMBUFP^); (* MAGIC PROC *)
5154  10    6:1   37    '[': BEGIN COMMENTER(']'); GOTO 1 END;
```

```
5155  10    6:1    44    '(': BEGIN
5156  10    6:3    44          IF SYMBUFP^[SYMCURSOR+1]='*' THEN
5157  10    6:4    53              BEGIN
5158  10    6:5    53                  SYMCURSOR := SYMCURSOR+1;
5159  10    6:5    58                  COMMENTER('*');
5160  10    6:5    61                  SYMCURSOR := SYMCURSOR+1;
5161  10    6:5    66                  GOTO 1; (* GET ANOTHER TOKEN *)
5162  10    6:4    68              END
5163  10    6:3    68            ELSE
5164  10    6:4    70              SY := LPARENT;
5165  10    6:2    73        END;
5166  10    6:1    75    ')': SY := RPARENT;
5167  10    6:1    80    ',': SY := COMMA;
5168  10    6:1    85    ' ',' ': BEGIN SYMCURSOR := SYMCURSOR+1; GOTO 1; END;
5169  10    6:1    94    '.': BEGIN
5170  10    6:3    94          IF SYMBUFP^[SYMCURSOR+1]='.'
5171  10    6:3    99            THEN
5172  10    6:4    03              BEGIN
5173  10    6:5    03                SYMCURSOR := SYMCURSOR+1;
5174  10    6:5    08                SY := COLON
5175  10    6:4    08              END
5176  10    6:3    11            ELSE
5177  10    6:4    13              SY := PERIOD;
5178  10    6:2    16        END;
5179  10    6:1    18    ':': IF SYMBUFP^[SYMCURSOR+1]='='
5180  10    6:2    23           THEN
5181  10    6:3    27             BEGIN
5182  10    6:4    27               SYMCURSOR := SYMCURSOR+1;
5183  10    6:4    32               SY := BECOMES;
5184  10    6:3    35             END
5185  10    6:2    35          ELSE
5186  10    6:3    37            SY := COLON;
5187  10    6:1    42    ';': SY := SEMICOLON;
5188  10    6:1    47    '^': SY := ARROW;
5189  10    6:1    52    '[': SY := LBRACK;
5190  10    6:1    57    ']': SY := RBRACK;
5191  10    6:1    62    '*': BEGIN SY := MULOP; OP := MUL END;
5192  10    6:1    70    '+': BEGIN SY := ADDOP; OP := PLUS END;
5193  10    6:1    78    '-': BEGIN SY := ADDOP; OP := MINUS END;
5194  10    6:1    86    '/': BEGIN SY := MULOP; OP := RDIV END;
5195  10    6:1    94    '<': BEGIN
```

```
5196  10    6:3    94          SY := RELOP;
5197  10    6:3    97          OP := LTOP;
5198  10    6:3    00          CASE SYMBUFP^[SYMCURSOR+1] OF
5199  10    6:3    07            '>': BEGIN
5200  10    6:5    07                   OP := NEOP;
5201  10    6:5    10                   SYMCURSOR := SYMCURSOR+1
5202  10    6:4    11                 END;
5203  10    6:3    17            '=': BEGIN
5204  10    6:5    17                   OP := LEOP;
5205  10    6:5    20                   SYMCURSOR := SYMCURSOR+1
5206  10    6:4    21                 END
5207  10    6:3    25          END;
5208  10    6:2    38        END;
5209  10    6:1    40    '=': BEGIN SY := RELOP; OP := EQOP END;
5210  10    6:1    48    '>': BEGIN
5211  10    6:3    48          SY := RELOP;
5212  10    6:3    51          IF SYMBUFP^[SYMCURSOR+1]='='
5213  10    6:3    56            THEN
5214  10    6:4    60              BEGIN
5215  10    6:5    60                OP := GEOP;
5216  10    6:5    63                SYMCURSOR := SYMCURSOR+1;
5217  10    6:4    68              END
5218  10    6:3    68            ELSE
5219  10    6:4    70              OP := GTOP;
5220  10    6:2    73        END
5221  10    6:1    73 END (* CASE SYMBUFP^[SYMCURSOR] OF *);
5222  10    6:1    12   IF SY=OTHERSY THEN
5223  10    6:2    17     IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
5224  10    6:3    24       BEGIN CHECKEND; GETSTMTLEV := TRUE; GOTO 1 END
5225  10    6:2    31     ELSE ERROR(400);
5226  10    6:1    38   SYMCURSOR := SYMCURSOR+1; (* NEXT CALL TALKS ABOUT NEXT TOKEN *)
5227  10    6:0    43 END (*INSYMBOL*) ;
5228  10    6:0    62
5229  10    6:0    62
5230  10    6:0    62 (*$I #5:PROCS.A.TEXT*)
5230  10    6:0    62 (*$I #5:PROCS.B.TEXT*)
5231  10    6:0    62   (*      COPYRIGHT  (C) 1979, REGENTS OF THE      *)
5232  10    6:0    62   (*      UNIVERSITY OF CALIFORNIA, SAN DIEGO      *)
5233  10    6:0    62
5234  10    7:D     1   PROCEDURE SEARCHSECTION(*FCP: CTP; VAR FCP1: CTP*);
5235  10    7:0     0   BEGIN
```

```
5236  10    7:1     0      IF FCP <> NIL THEN
5237  10    7:2     5        IF TREESEARCH(FCP,FCP1,ID) = 0 THEN (*NADA*)
5238  10    7:2    15        ELSE FCP1 := NIL
5239  10    7:1    18      ELSE FCP1 := NIL
5240  10    7:0    23    END (*SEARCHSECTION*) ;
5241  10    7:0    38
5242  10    8:0     1    PROCEDURE SEARCHID(*FIDCLS: SETOFIDS; VAR FCP: CTP*);
5243  10    8:0     3      LABEL 1; VAR LCP: CTP;
5244  10    8:0     0    BEGIN
5245  10    8:1     0      FOR DISX := TOP DOWNTO 0 DO
5246  10    8:2    12        BEGIN LCP := DISPLAY[DISX].FNAME;
5247  10    8:3    21          IF LCP <> NIL THEN
5248  10    8:4    26            IF TREESEARCH(LCP,LCP,ID) = 0 THEN
5249  10    8:5    37              IF LCP^.KLASS IN FIDCLS THEN GOTO 1
5250  10    8:5    47              ELSE
5251  10    8:6    49                IF PRTERR THEN ERROR(103)
5252  10    8:6    54                ELSE LCP := NIL
5253  10    8:4    58            ELSE LCP := NIL
5254  10    8:2    63        END;
5255  10    8:1    74      IF PRTERR THEN
5256  10    8:2    78        BEGIN ERROR(104);
5257  10    8:3    81          IF TYPES IN FIDCLS THEN LCP := UTYPPTR
5258  10    8:3    87          ELSE
5259  10    8:4    93            IF ACTUALVARS IN FIDCLS THEN LCP := UVARPTR
5260  10    8:4    99            ELSE
5261  10    8:5    05              IF FIELD IN FIDCLS THEN LCP := UFLDPTR
5262  10    8:5    11              ELSE
5263  10    8:6    17                IF KONST IN FIDCLS THEN LCP := UCSTPTR
5264  10    8:6    23                ELSE
5265  10    8:7    29                  IF PROC IN FIDCLS THEN LCP := UPRCPTR
5266  10    8:7    35                  ELSE LCP := UFCTPTR
5267  10    8:2    41        END;
5268  10    8:1    45 1:   FCP := LCP
5269  10    8:0    46    END (*SEARCHID*) ;
5270  10    8:0    62
5271  10    9:0     1    PROCEDURE GETBOUNDS(*FSP: STP; VAR FMIN,FMAX: INTEGER*);
5272  10    9:0     0    BEGIN
5273  10    9:1     0      WITH FSP^ DO
5274  10    9:2     3        IF FORM = SUBRANGE THEN
5275  10    9:3     9          BEGIN FMIN := MIN.IVAL; FMAX := MAX.IVAL END
5276  10    9:2    17        ELSE
```

```
5277  10    9:3    19          BEGIN FMIN := 0;
5278  10    9:4    22            IF FSP = CHARPTR THEN FMAX := 255
5279  10    9:4    29            ELSE
5280  10    9:5    35              IF FSP^.FCONST <> NIL THEN
5281  10    9:6    41                FMAX := FSP^.FCONST^.VALUES.IVAL
5282  10    9:5    44              ELSE FMAX := 0
5283  10    9:3    50          END
5284  10    9:0    52    END (*GETBOUNDS*) ;
5285  10    9:0    64
5286  10   10:D     1    PROCEDURE SKIP(*FSYS: SETOFSYS*);
5287  10   10:0     0    BEGIN WHILE NOT(SY IN FSYS) DO INSYMBOL
5288  10   10:0    10    END (*SKIP*) ;
5289  10   10:0    28
5290  10   11:D     3    FUNCTION PAOFCHAR(*FSP: STP): BOOLEAN*);
5291  10   11:0     0    BEGIN PAOFCHAR := FALSE;
5292  10   11:1     3      IF FSP <> NIL THEN
5293  10   11:2     8        IF FSP^.FORM = ARRAYS THEN
5294  10   11:3    14          PAOFCHAR := FSP^.AISPACKD AND (FSP^.AELTYPE = CHARPTR)
5295  10   11:0    21    END (*PAOFCHAR*) ;
5296  10   11:0    36
5297  10   12:D     3    FUNCTION STRGTYPE(*FSP: STP) : BOOLEAN*);
5298  10   12:0     0    BEGIN STRGTYPE := FALSE;
5299  10   12:1     3      IF PAOFCHAR(FSP) THEN STRGTYPE := FSP^.AISSTRNG
5300  10   12:0    11    END (*STRGTYPE*) ;
5301  10   12:0    26
5302  10   13:D     3    FUNCTION DECSIZE(*I: INTEGER): INTEGER*);
5303  10   13:0     0    BEGIN DECSIZE := (I + 3) DIV 4 + 1 (*GROSS..MAXIMUM NEEDED SPACE*)
5304  10   13:0     5    (* BINARY FN. SHOULD BE ((I*332) DIV 100 + 1 + BITSPERWD) DIV BITSPERWD *)
5305  10   13:0     5    END (*DECSIZE*) ;
5306  10   13:0    22
5307  10   14:D     1    PROCEDURE CONSTANT(*FSYS: SETOFSYS; VAR FSP: STP; VAR FVALU: VALU*);
5308  10   14:D     7      VAR LSP: STP; LCP: CTP; SIGN: (NONE,POS,NEG);
5309  10   14:D    10          LVP: CSP;
5310  10   14:0     0    BEGIN LSP := NIL; FVALU.IVAL := 0;
5311  10   14:1     6      IF NOT(SY IN CONSTBEGSYS) THEN
5312  10   14:2    16        BEGIN ERROR(50); SKIP(FSYS+CONSTBEGSYS) END;
5313  10   14:1    34      IF SY IN CONSTBEGSYS THEN
5314  10   14:2    43        BEGIN
5315  10   14:3    43          IF SY = STRINGCONSTSY THEN
5316  10   14:4    48            BEGIN
5317  10   14:5    48              IF LGTH = 1 THEN LSP := CHARPTR
```

```
5318  10   14:5   54          ELSE
5319  10   14:6   60            BEGIN
5320  10   14:7   60              NEW(LSP,ARRAYS,TRUE,TRUE);
5321  10   14:7   65              LSP^ := STRGPTR^;
5322  10   14:7   70              LSP^.MAXLENG := LGTH;
5323  10   14:7   76              LSP^.INXTYPE := NIL;
5324  10   14:7   81              NEW(LVP);
5325  10   14:7   88              LVP^ := VAL.VALP^;
5326  10   14:7   94              VAL.VALP := LVP
5327  10   14:6   94            END;
5328  10   14:5   97          FVALU := VAL; INSYMBOL
5329  10   14:4   02        END
5330  10   14:3   04      ELSE
5331  10   14:4   06        BEGIN
5332  10   14:5   06          SIGN := NONE;
5333  10   14:5   09          IF (SY = ADDOP) AND (OP IN [PLUS,MINUS]) THEN
5334  10   14:6   20            BEGIN IF OP = PLUS THEN SIGN := POS ELSE SIGN := NEG;
5335  10   14:7   34              INSYMBOL
5336  10   14:6   34            END;
5337  10   14:5   36          IF SY = IDENT THEN
5338  10   14:6   41            BEGIN SEARCHID([KONST],LCP);
5339  10   14:7   49              WITH LCP^ DO
5340  10   14:8   52                BEGIN LSP := IDTYPE; FVALU := VALUES END;
5341  10   14:7   62              IF SIGN <> NONE THEN
5342  10   14:8   67                IF LSP = INTPTR THEN
5343  10   14:9   72                  BEGIN IF SIGN = NEG THEN
5344  10   14:1   77                    FVALU.IVAL := -FVALU.IVAL END
5345  10   14:8   82                ELSE
5346  10   14:9   84                  IF LSP = REALPTR THEN
5347  10   14:0   90                    BEGIN
5348  10   14:1   90                      IF SIGN = NEG THEN
5349  10   14:2   95                        BEGIN NEW(LVP,REEL);
5350  10   14:3   00                          LVP^.CCLASS := REEL;
5351  10   14:3   03                          LVP^.RVAL := -FVALU.VALP^.RVAL;
5352  10   14:3   15                          FVALU.VALP := LVP;
5353  10   14:2   18                        END
5354  10   14:0   18                    END
5355  10   14:9   18                  ELSE
5356  10   14:0   20                    IF COMPTYPES(LSP,LONGINTPTR) THEN
5357  10   14:1   29                      BEGIN
5358  10   14:2   29                        IF SIGN = NEG THEN
```

```
5359  10    14:3     34                              BEGIN NEW(LVP,LONG);
5360  10    14:4     39                                LVP^.CCLASS := LONG;
5361  10    14:4     42                                LVP^.LONGVAL[1] := - FVALU.VALP^.LONGVAL[1];
5362  10    14:4     62                                FVALU.VALP := LVP
5363  10    14:3     63                              END
5364  10    14:1     65                            END
5365  10    14:0     65                          ELSE ERROR(105);
5366  10    14:7     70                      INSYMBOL;
5367  10    14:6     72                    END
5368  10    14:5     72                  ELSE
5369  10    14:6     74                    IF SY = INTCONST THEN
5370  10    14:7     79                      BEGIN IF SIGN = NEG THEN VAL.IVAL := -VAL.IVAL;
5371  10    14:8     89                        LSP := INTPTR; FVALU := VAL; INSYMBOL
5372  10    14:7     97                      END
5373  10    14:6     99                    ELSE
5374  10    14:7     01                      IF SY = REALCONST THEN
5375  10    14:8     06                        BEGIN IF SIGN = NEG THEN
5376  10    14:0     11                                VAL.VALP^.RVAL := -VAL.VALP^.RVAL;
5377  10    14:9     24                          LSP := REALPTR; FVALU := VAL; INSYMBOL
5378  10    14:8     33                        END
5379  10    14:7     35                      ELSE
5380  10    14:8     37                        IF SY = LONGCONST THEN
5381  10    14:9     42                          BEGIN
5382  10    14:0     42                            IF SIGN = NEG THEN
5383  10    14:1     47                              BEGIN VAL.VALP^.LONGVAL[1] := - VAL.VALP^.LONGVAL[1];
5384  10    14:2     68                                NEW(LSP,LONGINT);
5385  10    14:2     73                                LSP^.SIZE := DECSIZE(LGTH);
5386  10    14:2     81                                LSP^.FORM := LONGINT;
5387  10    14:2     86                                FVALU := VAL;
5388  10    14:2     91                                INSYMBOL
5389  10    14:1     91                              END
5390  10    14:9     93                          END
5391  10    14:8     93                        ELSE
5392  10    14:9     95                          BEGIN ERROR(106); SKIP(FSYS) END
5393  10    14:4     07              END;
5394  10    14:3     07            IF NOT (SY IN FSYS) THEN
5395  10    14:4     17              BEGIN ERROR(6); SKIP(FSYS) END
5396  10    14:2     29            END;
5397  10    14:1     29      FSP := LSP
5398  10    14:0     30    END (*CONSTANT*) ;
5399  10    14:0     52
```

```
5400  10   15:0    3   FUNCTION COMPTYPES(*FSP1,FSP2: STP) : BOOLEAN*);
5401  10   15:0    5     VAR NXT1,NXT2: CTP; COMP: BOOLEAN;
5402  10   15:0    8       LTESTP1,LTESTP2 : TESTP;
5403  10   15:0    0   BEGIN
5404  10   15:1    0     IF FSP1 = FSP2 THEN COMPTYPES := TRUE
5405  10   15:1    5     ELSE
5406  10   15:2   10       IF (FSP1 = NIL) OR (FSP2 = NIL) THEN COMPTYPES := TRUE
5407  10   15:2   19       ELSE
5408  10   15:3   24         IF FSP1^.FORM = FSP2^.FORM THEN
5409  10   15:4   31           CASE FSP1^.FORM OF
5410  10   15:4   35             SCALAR:
5411  10   15:5   35               COMPTYPES := FALSE;
5412  10   15:4   40             SUBRANGE:
5413  10   15:5   40               COMPTYPES := COMPTYPES(FSP1^.RANGETYPE,
5414  10   15:5   42                                      FSP2^.RANGETYPE);
5415  10   15:4   52             POINTER:
5416  10   15:5   52                 BEGIN
5417  10   15:6   52                   COMP := FALSE; LTESTP1 := GLOBTESTP;
5418  10   15:6   59                   LTESTP2 := GLOBTESTP;
5419  10   15:6   63                   WHILE LTESTP1 <> NIL DO
5420  10   15:7   68                     WITH LTESTP1^ DO
5421  10   15:8   71                       BEGIN
5422  10   15:9   71                         IF (ELT1 = FSP1^.ELTYPE) AND
5423  10   15:9   76                           (ELT2 = FSP2^.ELTYPE) THEN COMP := TRUE;
5424  10   15:9   87                         LTESTP1 := LASTTESTP
5425  10   15:8   87                       END;
5426  10   15:6   93                   IF NOT COMP THEN
5427  10   15:7   97                     BEGIN NEW(LTESTP1);
5428  10   15:8   02                       WITH LTESTP1^ DO
5429  10   15:9   05                         BEGIN ELT1 := FSP1^.ELTYPE;
5430  10   15:0   11                           ELT2 := FSP2^.ELTYPE;
5431  10   15:0   15                           LASTTESTP := GLOBTESTP
5432  10   15:9   18                         END;
5433  10   15:8   21                       GLOBTESTP := LTESTP1;
5434  10   15:8   24                       COMP := COMPTYPES(FSP1^.ELTYPE,FSP2^.ELTYPE)
5435  10   15:7   28                     END;
5436  10   15:6   34                   COMPTYPES := COMP; GLOBTESTP := LTESTP2
5437  10   15:5   37                 END;
5438  10   15:4   42             LONGINT: COMPTYPES := TRUE;
5439  10   15:4   47             POWER:
5440  10   15:5   47               COMPTYPES := COMPTYPES(FSP1^.ELSET,FSP2^.ELSET);
```

```
5441  10   15:4    59              ARRAYS:
5442  10   15:5    59                BEGIN
5443  10   15:6    59                  COMP := COMPTYPES(FSP1^.AELTYPE,FSP2^.AELTYPE)
5444  10   15:6    63                          AND (FSP1^.AISPACKD = FSP2^.AISPACKD);
5445  10   15:6    76                  IF COMP AND FSP1^.AISPACKD THEN
5446  10   15:7    82                      COMP := (FSP1^.ELSPERWD = FSP2^.ELSPERWD)
5447  10   15:7    87                              AND (FSP1^.ELWIDTH = FSP2^.ELWIDTH)
5448  10   15:7    92                              AND (FSP1^.AISSTRNG = FSP2^.AISSTRNG);
5449  10   15:6    02                  IF COMP AND NOT STRGTYPE(FSP1) THEN
5450  10   15:7    12                    COMP := (FSP1^.SIZE = FSP2^.SIZE);
5451  10   15:6    19                  COMPTYPES := COMP;
5452  10   15:5    22                END;
5453  10   15:4    24              RECORDS:
5454  10   15:5    24                BEGIN NXT1 := FSP1^.FSTFLD; NXT2 := FSP2^.FSTFLD;
5455  10   15:6    32                  COMP := TRUE;
5456  10   15:6    35                  WHILE (NXT1 <> NIL) AND (NXT2 <> NIL) AND COMP DO
5457  10   15:7    46                    BEGIN COMP:=COMPTYPES(NXT1^.IDTYPE,NXT2^.IDTYPE);
5458  10   15:8    56                      NXT1 := NXT1^.NEXT; NXT2 := NXT2^.NEXT
5459  10   15:7    61                    END;
5460  10   15:6    66                  COMPTYPES := COMP AND (NXT1 = NIL) AND (NXT2 = NIL)
5461  10   15:6    74                               AND (FSP1^.RECVAR = NIL)
5462  10   15:6    79                               AND (FSP2^.RECVAR = NIL)
5463  10   15:5    84                END;
5464  10   15:4    89              FILES:
5465  10   15:5    89                COMPTYPES := COMPTYPES(FSP1^.FILTYPE,FSP2^.FILTYPE)
5466  10   15:4    93            END (*CASE*)
5467  10   15:3    24          ELSE (*FSP1^.FORM <> FSP2^.FORM*)
5468  10   15:4    26            IF FSP1^.FORM = SUBRANGE THEN
5469  10   15:5    32              COMPTYPES := COMPTYPES(FSP1^.RANGETYPE,FSP2)
5470  10   15:4    35            ELSE
5471  10   15:5    43              IF FSP2^.FORM = SUBRANGE THEN
5472  10   15:6    49                COMPTYPES := COMPTYPES(FSP1,FSP2^.RANGETYPE)
5473  10   15:5    52              ELSE COMPTYPES := FALSE
5474  10   15:0    60  END (*COMPTYPES*) ;
5475  10   15:0    90
5476  10   15:0    90
5477  10   16:D     1  PROCEDURE GENBYTE(*FBYTE: INTEGER*);
5478  10   16:0     0  BEGIN
5479  10   16:1     0    CODEP^[IC] := CHR(FBYTE); IC := IC+1
5480  10   16:0     5  END (*GENBYTE*) ;
5481  10   16:0    22
```

```
5482  10   17:D    1   PROCEDURE GENWORD(*FWORD: INTEGER*);
5483  10   17:D    2    VAR TEMP: CHAR;
5484  10   17:0    0   BEGIN
5485  10   17:1    0     IF ODD(IC) THEN IC := IC + 1;
5486  10   17:1    8     MOVELEFT(FWORD,CODEP^[IC],2);
5487  10   17:1   16     IF FLIPBYTES THEN
5488  10   17:2   20       BEGIN
5489  10   17:3   20         TEMP := CODEP^[IC];
5490  10   17:3   25         CODEP^[IC] := CODEP^[IC+1];
5491  10   17:3   33         CODEP^[IC+1] := TEMP
5492  10   17:2   37       END;
5493  10   17:1   39     IC := IC + 2
5494  10   17:0   40   END (*GENWORD*) ;
5495  10   17:0   56
5496  10   18:D    1 PROCEDURE WRITETEXT;
5497  10   18:0    0   BEGIN
5498  10   18:1    0     MOVELEFT(SYMBUFP^[SYMCURSOR],CODEP^[0],1024);
5499  10   18:1    9     IF USERINFO.ERRNUM = 0 THEN
5500  10   18:2   15       IF BLOCKWRITE(USERINFO.WORKCODE^,CODEP^[0],2,CURBLK) <> 2 THEN
5501  10   18:3   33         ERROR(402);
5502  10   18:1   38     CURBLK := CURBLK + 2
5503  10   18:0   41   END (*WRITETEXT*) ;
5504  10   18:0   58
5505  10   19:D    1   PROCEDURE WRITECODE(*FORCEBUF: BOOLEAN*);
5506  10   19:D    2     VAR CODEINX,LIC,I: INTEGER;
5507  10   19:0    0   BEGIN CODEINX := 0; LIC := IC;
5508  10   19:1    6     REPEAT
5509  10   19:2    6       I := 512-CURBYTE;
5510  10   19:2   15       IF I > LIC THEN I := LIC;
5511  10   19:2   23       MOVELEFT(CODEP^[CODEINX],DISKBUF[CURBYTE],I);
5512  10   19:2   34       CODEINX := CODEINX+I;
5513  10   19:2   39       CURBYTE := CURBYTE+I;
5514  10   19:2   47       IF (CURBYTE = 512) OR FORCEBUF THEN
5515  10   19:3   58         BEGIN
5516  10   19:4   58           IF USERINFO.ERRNUM = 0 THEN
5517  10   19:5   64             IF BLOCKWRITE(USERINFO.WORKCODE^,DISKBUF,1,CURBLK) <> 1 THEN
5518  10   19:6   84               ERROR(402);
5519  10   19:4   89           CURBLK := CURBLK+1; CURBYTE := 0
5520  10   19:3   97         END;
5521  10   19:2   01       LIC := LIC-I
5522  10   19:1   02     UNTIL LIC = 0;
```

```
5523  10   19:0    11   END (*WRITECODE*) ;
5524  10   19:0    26
5525  10   26:0     1   PROCEDURE FINISHSEG;
5526  10   26:0     1     VAR I: INTEGER;
5527  10   26:0     0   BEGIN IC := 0;
5528  10   26:1     3     FOR I := NEXTPROC-1 DOWNTO 1 DO
5529  10   26:2    17       IF PROCTABLE[I] = 0 THEN
5530  10   26:3    28         GENWORD(0)
5531  10   26:2    29       ELSE
5532  10   26:3    33         GENWORD(SEGINX+IC-PROCTABLE[I]);
5533  10   26:1    54     GENBYTE(SEG); GENBYTE(NEXTPROC-1);
5534  10   26:1    63     SEGTABLE[SEG].CODELENG := SEGINX+IC;
5535  10   26:1    74     WRITECODE(TRUE); SEGINX := 0; CODEINSEG := FALSE
5536  10   26:0    80   END (*FINISHSEG*) ;
5537  10   26:0    98
5538  10   26:0    98
5539  10   26:0    98 (*$I #5:PROCS.B.TEXT*)
5539  10   26:0    98 (*$I #5:BLOCK.TEXT*)
5540  10   26:0    98
5541  10   20:0     1 PROCEDURE BLOCK(*FSYS: SETOFSYS*);
5542  10   20:0     5 LABEL 1;
5543  10   20:0     5 VAR BFSYFOUND: BOOLEAN;
5544  10   20:0     6
5545  10   27:0     1   PROCEDURE FINDFORW(FCP: CTP);
5546  10   27:0     0     BEGIN
5547  10   27:1     0       IF FCP <> NIL THEN
5548  10   27:2     5         WITH FCP^ DO
5549  10   27:3     8           BEGIN
5550  10   27:4     8             IF KLASS IN [PROC,FUNC] THEN
5551  10   27:5    16               IF PFDECKIND = DECLARED THEN
5552  10   27:6    23                 IF PFKIND = ACTUAL THEN
5553  10   27:7    30                   IF FORWDECL THEN
5554  10   27:8    35                     BEGIN
5555  10   27:9    35                       USERINFO.ERRNUM := 117; WRITELN(OUTPUT);
5556  10   27:9    46                       WRITE(OUTPUT,NAME,' UNDEFINED')
5557  10   27:8    75                     END;
5558  10   27:4    75             FINDFORW(RLINK); FINDFORW(LLINK)
5559  10   27:3    81           END
5560  10   27:0    83     END (*FINDFORW*) ;
5561  10   27:0    96
5562  10   20:0     0   BEGIN (*BLOCK*)
```

```
5563  10   20:1    0      IF (NOSWAP) AND (STARTINGUP) THEN
5564  10   20:2    7        BEGIN
5565  10   20:3    7          BODYPART(FSYS,NIL);
5566  10   20:3   18          EXIT(BLOCK);
5567  10   20:2   22        END;
5568  10   20:1   22      IF (SY IN [UNITSY,SEPARATSY]) AND (NOT INMODULE) THEN
5569  10   20:2   42        BEGIN
5570  10   20:3   42          UNITPART(FSYS + [UNITSY,INTERSY,IMPLESY,ENDSY]);
5571  10   20:3   65          IF SY = PERIOD THEN EXIT(BLOCK)
5572  10   20:2   74        END;
5573  10   20:1   74      NEWBLOCK:=TRUE;
5574  10   20:1   77      REPEAT
5575  10   20:2   77        IF NOT NEWBLOCK THEN
5576  10   20:3   82          BEGIN
5577  10   20:4   82            OP := FALSE; STMTLEV := 0; IC := 0; LINEINFO := 0;
5578  10   20:4   94            IF (NOT SYSCOMP) OR (LEVEL>1) THEN FINDFORW(DISPLAY[TOP].FNAME);
5579  10   20:4   12            IF INMODULE THEN
5580  10   20:5   16              IF TOS^.PREVLEXSTACKP^.DFPROCP = OUTERBLOCK THEN
5581  10   20:6   26                IF (SY = ENDSY) THEN
5582  10   20:7   31                  BEGIN FINISHSEG; EXIT(BLOCK) END
5583  10   20:6   37                ELSE IF (SY = BEGINSY) THEN
5584  10   20:8   44                  BEGIN ERROR(13); FINISHSEG; EXIT(BLOCK) END;
5585  10   20:4   53            IF SY = BEGINSY THEN INSYMBOL ELSE ERROR(17);
5586  10   20:4   65            REPEAT
5587  10   20:5   65              BODYPART(FSYS + [CASESY] - [ENDSY], TOS^.DFPROCP);
5588  10   20:5   91              BFSYFOUND := (SY = TOS^.BFSY) OR (INMODULE AND (SY = ENDSY));
5589  10   20:5   05              IF NOT BFSYFOUND THEN
5590  10   20:6   09                BEGIN
5591  10   20:7   09                  IF TOS^.BFSY = SEMICOLON THEN
5592  10   20:8   16                    ERROR(14)  (*SEMICOLON EXPECTED*)
5593  10   20:7   17                  ELSE  ERROR(6);  (* PERIOD EXPECTED *)
5594  10   20:7   24                  SKIP(FSYS + [TOS^.BFSY]);
5595  10   20:7   38                  BFSYFOUND := (SY = TOS^.BFSY) OR (INMODULE AND (SY = ENDSY))
5596  10   20:6   49                END
5597  10   20:4   52            UNTIL (BFSYFOUND) OR (SY IN BLOCKBEGSYS);
5598  10   20:4   63            IF NOT BFSYFOUND THEN
5599  10   20:5   67              BEGIN
5600  10   20:6   67                IF TOS^.BFSY = SEMICOLON THEN ERROR(14)
5601  10   20:6   75                ELSE ERROR(6); (*PERIOD EXPECTED*)
5602  10   20:6   82                DECLARATIONPART(FSYS);
5603  10   20:5   92              END
```

```
5604  10   20:4   92              ELSE
5605  10   20:5   94                BEGIN
5606  10   20:6   94                  IF SY = SEMICOLON THEN INSYMBOL;
5607  10   20:6   01                  IF (NOT(SY IN [BEGINSY,PROCSY,FUNCSY,PROGSY])) AND
5608  10   20:6   13                     (TOS^.BFSY = SEMICOLON) THEN
5609  10   20:7   21                    IF NOT (INMODULE AND (SY = ENDSY)) THEN
5610  10   20:8   30                      BEGIN
5611  10   20:9   30                        ERROR(6); SKIP(FSYS);
5612  10   20:9   42                        DECLARATIONPART(FSYS);
5613  10   20:8   52                      END
5614  10   20:7   52                    ELSE GOTO 1
5615  10   20:6   56                  ELSE
5616  10   20:7   58          1:        BEGIN
5617  10   20:8   58                    WITH TOS^ DO
5618  10   20:9   62                      BEGIN
5619  10   20:0   62                        IF DFPROCP <> NIL THEN
5620  10   20:1   68                          DFPROCP^.INSCOPE:=FALSE;
5621  10   20:0   74                        IF ISSEGMENT THEN
5622  10   20:1   79                          BEGIN
5623  10   20:2   79                            IF CODEINSEG THEN FINISHSEG;
5624  10   20:2   85                            IF DLINKERINFO AND (LEVEL = 1) THEN
5625  10   20:3   94                              BEGIN SEGTABLE[SEG].SEGKIND := 2;
5626  10   20:4   04                                WRITELINKERINFO(TRUE)
5627  10   20:3   05                              END
5628  10   20:2   08                            ELSE
5629  10   20:3   10                              IF CLINKERINFO THEN
5630  10   20:4   14                                BEGIN SEGTABLE[SEG].SEGKIND := 2;
5631  10   20:5   24                                  WRITELINKERINFO(FALSE)
5632  10   20:4   25                                END;
5633  10   20:2   28                            NEXTPROC:=SOLDPROC;
5634  10   20:2   32                            SEG:=DOLDSEG;
5635  10   20:1   36                          END;
5636  10   20:0   36                        LEVEL:=DOLDLEV;
5637  10   20:0   40                        TOP:=DOLDTOP;
5638  10   20:0   44                        LC:=DLLC;
5639  10   20:0   48                        CURPROC:=POLDPROC;
5640  10   20:9   52                      END;
5641  10   20:8   52                    RELEASE(TOS^.DMARKP);
5642  10   20:8   58                    TOS:=TOS^.PREVLEXSTACKP;
5643  10   20:8   64                    NEWBLOCK:=(SY IN [PROCSY,FUNCSY,PROGSY]);
5644  10   20:7   78                  END
```

```
5645  10    20:5     78              END
5646  10    20:3     78            END
5647  10    20:2     78          ELSE
5648  10    20:3     80            BEGIN DECLARATIONPART(FSYS);
5649  10    20:4     90              IF LEVEL = 0 THEN
5650  10    20:5     96                IF SY IN [UNITSY,SEPARATSY] THEN
5651  10    20:6     12                  BEGIN
5652  10    20:7     12                    UNITPART(FSYS + [UNITSY,INTERSY,IMPLESY,ENDSY]);
5653  10    20:7     35                    IF SY IN [PROCSY,FUNCSY,PROGSY] THEN DECLARATIONPART(FSYS)
5654  10    20:6     55                  END
5655  10    20:3     58            END;
5656  10    20:1     58        UNTIL TOS = NIL;
5657  10    20:1     64        FINISHSEG;
5658  10    20:0     66  END (*BLOCK*) ;
5659  10    20:0     86
5660  10     1:0      0 BEGIN (* PASCALCOMPILER *)
5661  10     1:1      0   COMPINIT;
5662  10     1:1     55   TIME(LGTH,LOWTIME);
5663  10     1:1     61   BLOCK(BLOCKBEGSYS+STATBEGSYS-[CASESY]);
5664  10     1:1     84   IF SY <> PERIOD THEN ERROR(21);
5665  10     1:1     92   IF LIST THEN
5666  10     1:2     96     BEGIN SCREENDOTS := SCREENDOTS+1;
5667  10     1:3     02       SYMBUFP^[SYMCURSOR] := CHR(EOL);
5668  10     1:3     06       SYMCURSOR := SYMCURSOR+1;
5669  10     1:3     11       PRINTLINE
5670  10     1:2     11     END;
5671  10     1:1     13   USERINFO.ERRBLK := 0;
5672  10     1:1     18   TIME(LGTH,STARTDOTS); LOWTIME := STARTDOTS-LOWTIME;
5673  10     1:1     31   UNITWRITE(3,IC,7);
5674  10     1:1     40   IF DLINKERINFO OR CLINKERINFO THEN
5675  10     1:2     47     BEGIN SEGTABLE[SEG].SEGKIND := 1;
5676  10     1:3     57       WRITELINKERINFO(TRUE)
5677  10     1:2     58     END;
5678  10     1:1     61   CLOSE(LP,LOCK);
5679  10     1:1     68   IF NOISY THEN WRITELN(OUTPUT);
5680  10     1:1     78   WRITE(OUTPUT,SCREENDOTS,' LINES');
5681  10     1:1     03   IF LOWTIME > 0 THEN
5682  10     1:2     09     WRITE(OUTPUT,', ',(LOWTIME+30) DIV 60,' SECS, ',
5683  10     1:2     51         ROUND((3600/LOWTIME)*SCREENDOTS),' LINES/MIN');
5684  10     1:1     92   IF NOISY THEN
5685  10     1:2     96     BEGIN
```

```
5686  10    1:3    96        WRITELN(OUTPUT);
5687  10    1:3    02        WRITE(OUTPUT,'SMALLEST AVAILABLE SPACE = ',SMALLESTSPACE,' WORDS');
5688  10    1:2    64      END;
5689  10    1:1    64    IC := 0;
5690  10    1:1    67    FOR SEG := 0 TO MAXSEG DO
5691  10    1:2    81      WITH SEGTABLE[SEG] DO
5692  10    1:3    90        BEGIN GENWORD(DISKADDR); GENWORD(CODELENG) END;
5693  10    1:1    09    FOR SEG := 0 TO MAXSEG DO
5694  10    1:2    23      WITH SEGTABLE[SEG] DO
5695  10    1:3    32        FOR LGTH := 1 TO 8 DO
5696  10    1:4    47          GENBYTE(ORD(SEGNAME[LGTH]));
5697  10    1:1    74    FOR SEG := 0 TO MAXSEG DO GENWORD(SEGTABLE[SEG].SEGKIND);
5698  10    1:1    04    FOR SEG := 0 TO MAXSEG DO GENWORD(SEGTABLE[SEG].TEXTADDR);
5699  10    1:1    34    FOR LGTH := 1 TO 80 DO
5700  10    1:2    49      IF COMMENT <> NIL THEN GENBYTE(ORD(COMMENT^[LGTH])) ELSE GENBYTE(0);
5701  10    1:1    77    FOR LGTH := 1 TO 256 - 8*(MAXSEG + 1) - 40 DO GENWORD(0);
5702  10    1:1    13    CURBLK := 0; CURBYTE := 0; WRITECODE(TRUE)
5703  10    1:0    22  END (* PASCALCOMPILER *) ;
5704  10    1:0    78
5705   0    1:0     0  BEGIN (* SYSTEM *)
5706   0    1:0     0  END.
```

AT THE TIME OF THE PRINTING OF THIS BOOK
THE BASIC COMPILER WAS NOT LISTED.

THE BASIC COMPILER WILL BE AVAILABLE IN
A SUPPLIMENTAL BOOK AT SOME TIME IN THE
FUTURE AT ADDITIONAL COST.

THANK YOU FOR YOUR PATIENCE IN THE MATTER. ED.

```
  1    1     1:0     1 (*$L PRINTER:*)
  1    1     1:0     1 [$I LINK0 ]
  2    1     1:0     1     (*****************************************************************)
  3    1     1:0     1     (*                                                               *)
  4    1     1:0     1     (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
  5    1     1:0     1     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
  6    1     1:0     1     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
  7    1     1:0     1     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
  8    1     1:0     1     (*                                                               *)
  9    1     1:0     1     (*****************************************************************)
 10    1     1:0     1
 11    1     1:0     1 [$S+,U-,R+
 12    1     1:0     1
 13    1     1:0     1
 14    1     1:0     1          UCSD  PASCAL  SYSTEM
 15    1     1:0     1            PROGRAM  LINKER
 16    1     1:0     1
 17    1     1:0     1            (VERSION I.5F)
 18    1     1:0     1
 19    1     1:0     1          WRITTEN SUMMER '78 BY
 20    1     1:0     1          ROGER T. SUMNER, IIS
 21    1     1:0     1
 22    1     1:0     1          COPYRIGHT (C) 1978, REGENTS OF
 23    1     1:0     1          THE UNIVERSITY OF CALIFORNIA
 24    1     1:0     1
 25    1     1:0     1      ALL HOPE ABANDON YE WHO ENTER HERE
 26    1     1:0     1                           -DANTE
 27    1     1:0     1
 28    1     1:0     1 ]
 29    1     1:0     1
 30    0     1:0     1 PROGRAM SYSTEMLEVEL;
 31    0     1:0     1
 32    0     1:0     1 CONST
 33    0     1:0     1     SYSPROG = 4;
 34    0     1:0     1
 35    0     1:0     1 VAR
 36    0     1:0     1     SYSCOM: ^INTEGER;
 37    0     1:0     2     GFILES: ARRAY [0..5] OF INTEGER;
 38    0     1:0     8     USERINFO: RECORD
 39    0     1:0     8                 FILLER: ARRAY [0..4] OF INTEGER;
 40    0     1:0     8                 SLOWTERM, STUPID: BOOLEAN;
```

```
41   0    1:D     8                ALTMODE: CHAR;
42   0    1:D     5                GOTSYM, GOTCODE: BOOLEAN;
43   0    1:D     8                WORKVID, SYMVID, CODEVID: STRING[7];
44   0    1:D     8                WORKTID, SYMTID, CODETID: STRING[15]
45   0    1:D     8              END;
46   0    1:D    54    FILLER: ARRAY [0..4] OF INTEGER;
47   0    1:D    59    SYVID, DKVID: STRING[7];
48   0    1:D    67    JUNK1, JUNK2: INTEGER;
49   0    1:D    69    CMDSTATE: INTEGER;
50   0    1:D    70
51   0    1:D    70 [
52   0    1:D    70 *  THE LINKER IS MADE UP OF THREE PHASES:
53   0    1:D    70 *    PHASE1 WHICH OPEN ALL INPUT FILES, READS UP SEG TABLES
54   0    1:D    70 *           FROM THEM AND DECIDES WHICH SEGMENTS ARE TO BE
55   0    1:D    70 *           LINKED INTO THE FINAL CODE FILE.
56   0    1:D    70 *    PHASE2 READS THE LINKER INFO FOR EACH SEGMENT THAT IS
57   0    1:D    70 *           GOING TO BE USED, EITHER TO SELECT SEP PROCS FROM
58   0    1:D    70 *           OR COPY WITH MODIFICATIONS INTO OUTPUT CODE.
59   0    1:D    70 *           THE MAIN SYMBOL TREES ARE BUILT HERE, ONE FOR EACH
60   0    1:D    70 *           CODE SEGMENT.
61   0    1:D    70 *    PHASE3 DOES THE CRUNCHING OF CODE SEGMENTS INTO THEIR
62   0    1:D    70 *           FINAL FORM BY FIGURING OUT THE PROCS THAT NEED TO
63   0    1:D    70 *           BE LINKED IN, RESOLVES ALL REFERENCES (PUBLREF,
64   0    1:D    70 *           GLOBREF, ETC), PATCHES THE CODE POINTED TO BY THEIR
65   0    1:D    70 *           REFLISTS, AND WRITES THE FINAL CODE SEG(S).
66   0    1:D    70 ]
67   0    1:D    70
68   1    1:D     1 SEGMENT PROCEDURE LINKER(III, JJJ: INTEGER);
69   1    1:D     3
70   1    1:D     3 CONST
71   1    1:D     3     HEADER = 'LINKER [I.5F]';
72   1    1:D     3
73   1    1:D     3     MAXSEG = 15;          [ MAX CODE SEG # IN CODE FILES ]
74   1    1:D     3     MAXSEG1 = 16;         [ MAXSEG+1, USEFUL FOR LOOP VARS ]
75   1    1:D     3     MASTERSEG = 1;        [ USERHOST SEGMENT NUMBER # ]
76   1    1:D     3     FIRSTSEG =  7;        [ FIRST LINKER ASSIGNABLE SEG # ]
77   1    1:D     3     MAXFILE = 7;          [ NUMBER OF LIB FILES WE CAN USE ]
78   1    1:D     3     MAXLC = MAXINT;       [ MAX COMPILER ASSIGNED ADDRESS ]
79   1    1:D     3     MAXIC = 20000;        [ MAX NUMBER BYTES OF CODE PER PROC ]
80   1    1:D     3     MAXPROC = 160;        [ MAX LEGAL PROCEDURE NUMBER ]
81   1    1:D     3     MSDELTA = 12;         [ MARK STACK SIZE FOR PUB/PRIV FIXUP ]
```

```
 82   1     1:D     3
 83   1     1:D     3 TYPE
 84   1     1:D     3
 85   1     1:D     3     [ SUBRANGES ]
 86   1     1:D     3     [ --------- ]
 87   1     1:D     3
 88   1     1:D     3     SEGRANGE = 0..MAXSEG;          [ SEG TABLE SUBSCRIPT TYPE ]
 89   1     1:D     3     SEGINDEX = 0..MAXSEG1;         [ WISH WE HAD CONST EXPRESSIONS! ]
 90   1     1:D     3     LCRANGE = 1..MAXLC;            [ BASE OFFSETS A LA P-CODE ]
 91   1     1:D     3     ICRANGE = 0..MAXIC;            [ LEGAL LENGTH FOR PROC/FUNC CODE ]
 92   1     1:D     3     PROCRANGE = 1..MAXPROC;        [ LEGIT PROCEDURE NUMBERS ]
 93   1     1:D     3
 94   1     1:D     3     [ MISCELLANEOUS ]
 95   1     1:D     3     [ ------------- ]
 96   1     1:D     3
 97   1     1:D     3     ALPHA = PACKED ARRAY [0..7] OF CHAR;
 98   1     1:D     3     DISKBLOCK = PACKED ARRAY [0..511] OF 0..255;
 99   1     1:D     3     CODEFILE = FILE;               [ TRICK COMPILER TO GET ^FILE ]
100   1     1:D     3     FILEP = ^CODEFILE;
101   1     1:D     3     CODEP = ^DISKBLOCK;            [ SPACE MANAGEMENT...NON-PASCAL KLUDGE ]
102   1     1:D     3
103   1     1:D     3     [ LINK INFO STRUCTURES ]
104   1     1:D     3     [ ---- ---- ---------- ]
105   1     1:D     3
106   1     1:D     3     PLACEP = ^PLACEREC;            [ POSITION IN SOURCE SEG ]
107   1     1:D     3     PLACEREC = RECORD
108   1     1:D     3                  SRCBASE, DESTBASE: INTEGER;
109   1     1:D     3                  LENGTH: ICRANGE
110   1     1:D     3                END [ PLACEREC ] ;
111   1     1:D     3
112   1     1:D     3     REFP = ^REFNODE;               [ IN-CORE VERSION OF REF LISTS ]
113   1     1:D     3     REFNODE = RECORD
114   1     1:D     3                 NEXT: REFP;
115   1     1:D     3                 REFS: ARRAY [0..7] OF INTEGER;
116   1     1:D     3               END [ REFNODE ] ;
117   1     1:D     3
118   1     1:D     3     LITYPES = (EOFMARK,            [ END-OF-LINK-INFO MARKER ]
119   1     1:D     3                [ EXT REF TYPES, DESIGNATES      ]
120   1     1:D     3                [ FIELDS TO BE UPDATED BY LINKER ]
121   1     1:D     3                UNITREF,           [ REFS TO INVISIBLY USED UNITS (ARCHAIC?) ]
122   1     1:D     3                GLOBREF,           [ REFS TO EXTERNAL GLOBAL ADDRS ]
```

```
123   1     1:0     3                PUBLREF,          [ REFS TO BASE LEV VARS IN HOST ]
124   1     1:0     3                PRIVREF,          [ REFS TO BASE VARS, ALLOCATED BY LINKER ]
125   1     1:0     3                CONSTREF,         [ REFS TO HOST BASE LEV CONSTANT ]
126   1     1:0     3                    [ DEFINING TYPES, GIVES      ]
127   1     1:0     3                    [ LINKER VALUES TO FIX REFS  ]
128   1     1:0     3                GLOBDEF,          [ GLOBAL ADDR LOCATION ]
129   1     1:0     3                PUBLDEF,          [ BASE VAR LOCATION ]
130   1     1:0     3                CONSTDEF,         [ BASE CONST DEFINITION ]
131   1     1:0     3                    [ PROC/FUNC INFO, ASSEM ]
132   1     1:0     3                    [ TO PASCAL AND PASCAL  ]
133   1     1:0     3                    [ TO PASCAL INTERFACE   ]
134   1     1:0     3                EXTPROC,          [ EXTERNAL PROC TO BE LINKED INTO PASCAL ]
135   1     1:0     3                EXTFUNC,          [    "     FUNC "  "    "    "      "    ]
136   1     1:0     3                SEPPROC,          [ SEPARATE PROC DEFINITION RECORD ]
137   1     1:0     3                SEPFUNC,          [   "      FUNC     "         "    ]
138   1     1:0     3                SEPPREF,          [ PASCAL REF TO A SEP PROC ]
139   1     1:0     3                SEPFREF);         [   "    REF TO A SEP FUNC ]
140   1     1:0     3
141   1     1:0     3           LISET = SET OF LITYPES;
142   1     1:0     3           OPFORMAT = (WORD, BYTE, BIG);       [ INSTRUCTION OPERAND FIELD FORMATS ]
143   1     1:0     3
144   1     1:0     3           LIENTRY = RECORD    [ FORMAT OF LINK INFO RECORDS ]
145   1     1:0     3                       NAME: ALPHA;
146   1     1:0     3                       CASE LITYPE: LITYPES OF
147   1     1:0     3                         SEPPREF,
148   1     1:0     3                         SEPFREF,
149   1     1:0     3                         UNITREF,
150   1     1:0     3                         GLOBREF,
151   1     1:0     3                         PUBLREF,
152   1     1:0     3                         PRIVREF,
153   1     1:0     3                         CONSTREF:
154   1     1:0     3                               (FORMAT: OPFORMAT;      [ HOW TO DEAL WITH THE REFS ]
155   1     1:0     3                                NREFS: INTEGER;        [ WORDS FOLLOWING WITH REFS ]
156   1     1:0     3                                NWORDS: LCRANGE;       [ SIZE OF PRIVATE OR NPARAMS ]
157   1     1:0     3                                REFLIST: REFP);        [ LIST OF REFS AFTER READ IN ]
158   1     1:0     3                         EXTPROC,
159   1     1:0     3                         EXTFUNC,
160   1     1:0     3                         SEPPROC,
161   1     1:0     3                         SEPFUNC:
162   1     1:0     3                               (SRCPROC: PROCRANGE;    [ THE PROCNUM IN SOURCE SEG ]
163   1     1:0     3                                NPARAMS: INTEGER;      [ WORDS PASSED/EXPECTED ]
```

```
164   1    1:D     3                    PLACE: PLACEP);          [ POSITION IN SOURCE/DEST SEG ]
165   1    1:D     3                GLOBDEF:
166   1    1:D     3                      (HOMEPROC: PROCRANGE;  [ WHICH PROC IT OCCURS IN ]
167   1    1:D     3                       ICOFFSET: ICRANGE);   [ ITS BYTE OFFSET IN PCODE ]
168   1    1:D     3                PUBLDEF:
169   1    1:D     3                      (BASEOFFSET: LCRANGE); [ COMPILER ASSIGN WORD OFFSET ]
170   1    1:D     3                CONSTDEF:
171   1    1:D     3                      (CONSTVAL: INTEGER);   [ USERS DEFINED VALUE ]
172   1    1:D     3                EOFMARK:
173   1    1:D     3                      (NEXTLC: LCRANGE)      [ PRIVATE VAR ALLOC INFO ]
174   1    1:D     3              END [ LIENTRY ] ;
175   1    1:D     3
176   1    1:D     3     [ SYMBOL TABLE ITEMS ]
177   1    1:D     3     [ ------ ----- ----- ]
178   1    1:D     3
179   1    1:D     3     SYMP = ^SYMBOL;
180   1    1:D     3     SYMBOL = RECORD
181   1    1:D     3                LLINK, RLINK,            [ BINARY SUBTREES FOR DIFF NAMES ]
182   1    1:D     3                SLINK: SYMP;             [ SAME NAME, DIFF LITYPES ]
183   1    1:D     3                ENTRY: LIENTRY           [ ACTUAL ID INFORMATION ]
184   1    1:D     3              END [ SYMBOL ] ;
185   1    1:D     3
186   1    1:D     3     [ SEGMENT INFORMATION ]
187   1    1:D     3     [ ------- ----------- ]
188   1    1:D     3
189   1    1:D     3     SEGKINDS =(LINKED,           [ NO WORK NEEDED, EXECUTABLE AS IS ]
190   1    1:D     3                HOSTSEG,          [ PASCAL HOST PROGRAM OUTER BLOCK  ]
191   1    1:D     3                SEGPROC,          [ PASCAL SEGMENT PROCEDURE, NOT HOST ]
192   1    1:D     3                UNITSEG,          [ LIBRARY UNIT OCCURANCE/REFERENCE ]
193   1    1:D     3                SEPRTSEG);        [ LIBRARY SEPARATE PROC/FUNC TLA SEGMENT ]
194   1    1:D     3
195   1    1:D     3     FINFOP = ^FILEINFOREC;       [ FORWARD TYPE DEC ]
196   1    1:D     3
197   1    1:D     3     SEGP = ^SEGREC;              [ THIS STRUCTURE PROVIDES ACCESS TO ALL ]
198   1    1:D     3     SEGREC = RECORD              [ INFO FOR SEGS TO BE LINKED TO/FROM    ]
199   1    1:D     3                SRCFILE: FINFOP;          [ SOURCE FILE OF SEGMENT ]
200   1    1:D     3                SRCSEG: SEGRANGE;         [ SOURCE FILE SEG # ]
201   1    1:D     3                SYMTAB: SYMP;             [ SYMBOL TABLE TREE ]
202   1    1:D     3                CASE SEGKIND: SEGKINDS OF
203   1    1:D     3                  SEPRTSEG:
204   1    1:D     3                        (NEXT: SEGP)     [ USED FOR LIBRARY SEP SEG LIST ]
```

```
205   1    1:D    3              END [ SEGREC ] ;
206   1    1:D    3
207   1    1:D    3    [ HOST/LIB FILE ACCESS INFO ]
208   1    1:D    3    [ ---- --- ---- ------ ---- ]
209   1    1:D    3
210   1    1:D    3    I5SEGTBL = RECORD   [ FIRST FULL BLOCK OF ALL CODE FILES ]
211   1    1:D    3                 DISKINFO: ARRAY [SEGRANGE] OF
212   1    1:D    3                             RECORD
213   1    1:D    3                               CODELENG, CODEADDR: INTEGER
214   1    1:D    3                             END [ DISKINFO ] ;
215   1    1:D    3                 SEGNAME: ARRAY [SEGRANGE] OF ALPHA;
216   1    1:D    3                 SEGKIND: ARRAY [SEGRANGE] OF SEGKINDS;
217   1    1:D    3                 FILLER: ARRAY [0..143] OF INTEGER
218   1    1:D    3               END [ I5SEGTBL ] ;
219   1    1:D    3
220   1    1:D    3    FILEKIND = (USERHOST, USERLIB, SYSTEMLIB);
221   1    1:D    3
222   1    1:D    3    FILEINFOREC = RECORD
223   1    1:D    3                    NEXT: FINFOP;        [ LINK TO NEXT FILE THATS OPEN ]
224   1    1:D    3                    CODE: FILEP;         [ POINTER TO PASCAL FILE...SNEAKY! ]
225   1    1:D    3                    FKIND: FILEKIND;     [ USED TO VALIDATE THE SEGKINDS ]
226   1    1:D    3                    SEGTBL: I5SEGTBL     [ DISK SEG TABLE W/ SOURCE INFO ]
227   1    1:D    3                  END [ FILEINFOREC ] ;
228   1    1:D    3
229   1    1:D    3
230   1    1:D    3  VAR
231   1    1:D    3      HOSTFILE,              [ HOST FILE INFO PTR, ITS NEXT = LIBFILES ]
232   1    1:D    3      LIBFILES: FINFOP;      [ LIST OF LIB FILES, USER AND SYSTEM ]
233   1    1:D    5
234   1    1:D    5      SEPLIST: SEGP;         [ LIST OF SEP SEGS TO SEARCH THROUGH ]
235   1    1:D    6      REFLITYPES: LISET;     [ THOSE LITYPES WITH REF LISTS ]
236   1    1:D    7
237   1    1:D    7      TALKATIVE,
238   1    1:D    7      USEWORKFILE: BOOLEAN;
239   1    1:D    9
240   1    1:D    9      ERRCOUNT: INTEGER;
241   1    1:D   10      HEAPBASE: ^INTEGER;
242   1    1:D   11
243   1    1:D   11      HOSTSP: SEGP;                          [ PTR TO HOST PROG OUTER BLOCK ]
244   1    1:D   12      NEXTBASELC: LCRANGE;                   [ NEXT BASE OFFSET FOR PRIVATE ALLOC ]
245   1    1:D   13      SEGINFO: ARRAY [SEGRANGE] OF SEGP;     [ SEG IS AVAILABLE IF NIL ]
```

```
246   1    1:D    29      NEXTSEG: SEGINDEX;                          [ NEXT SLOT IN SEGINFO AVAILABLE ]
247   1    1:D    30
248   1    1:D    30      MAPNAME: STRING[40];
249   1    1:D    51
250   1    1:D    51      F0, F1, F2, F3,
251   1    1:D    51      F4, F5, F6, F7,                             [ INPUT FILES WITH LURKING PNTRS ]
252   1    1:D    51      CODE: CODEFILE;                             [ OUTPUT CODE FILE, *SYSTEM.WRK.CODE ]
253   1    1:D    11
254   1    1:D    11      FLIPPED: BOOLEAN;                           [ ARE FILES BYTE-FLIPPED? ]
255   1    1:D    12 [
256   1    1:D    12 *  PRINT AN ERROR MESSAGE AND BUMP
257   1    1:D    12 *  THE ERROR COUNTER.
258   1    1:D    12 ]
259   1    1:D    12
260   1    2:D     1 PROCEDURE ERROR(MSG: STRING);
261   1    2:D    43   VAR CH: CHAR;
262   1    2:0     0 BEGIN
263   1    2:1     0   WRITELN(MSG);
264   1    2:1    20   REPEAT
265   1    2:2    20     WRITELN('TYPE <SP>(CONTINUE), <ESC>(TERMINATE)');
266   1    2:2    73     READ(KEYBOARD, CH);
267   1    2:2    81     IF CH = USERINFO.ALTMODE THEN
268   1    2:3    89       EXIT(LINKER)
269   1    2:1    93   UNTIL CH = ' ';
270   1    2:1    99   ERRCOUNT := ERRCOUNT+1
271   1    2:0    00 END [ ERROR ] ;
272   1    2:0    18
273   1    3:D     1 PROCEDURE BYTESWAP(VAR WORD: INTEGER);
274   1    3:D     2   VAR TEMP1,TEMP2: PACKED RECORD
275   1    3:D     2                 CASE BOOLEAN OF
276   1    3:D     2                   TRUE: (VAL: INTEGER);
277   1    3:D     2                   FALSE: (LOWBYTE: 0..255;
278   1    3:D     2                           HIGHBYTE: 0..255)
279   1    3:D     2               END;
280   1    3:0     0 BEGIN
281   1    3:1     0   TEMP1.VAL := WORD;
282   1    3:1     4   TEMP2.LOWBYTE := TEMP1.HIGHBYTE;
283   1    3:1    19   TEMP2.HIGHBYTE := TEMP1.LOWBYTE;
284   1    3:1    34   WORD := TEMP2.VAL;
285   1    3:0    37 END;
286   1    3:0    50
```

```
287   1    3:0    50 [
288   1    3:0    50 *  ROUTINES TO ACCESS OBJECT CODE SEGMENTS.  THERE
289   1    3:0    50 *  IS SUBTLE BUSINESS INVOLVING BYTE FLIPPING WITH
290   1    3:0    50 *  THE 16-BIT OPERATIONS.
291   1    3:0    50 ]
292   1    3:0    50 [$R-]
293   1    3:0    50
294   1    4:0     3 FUNCTION FETCHBYTE(CP: CODEP; OFFSET: INTEGER): INTEGER;
295   1    4:0     0 BEGIN
296   1    4:1     0   FETCHBYTE := CP^[OFFSET]
297   1    4:0     2 END [ FETCHBYTE ] ;
298   1    4:0    18
299   1    5:0     3 FUNCTION FETCHWORD(CP: CODEP; OFFSET: INTEGER): INTEGER;
300   1    5:0     5   VAR I: INTEGER;
301   1    5:0     0 BEGIN
302   1    5:1     0   MOVELEFT(CP^[OFFSET], I, 2);
303   1    5:1     8   [ BYTE SWAP I ]
304   1    5:1     8   IF FLIPPED THEN BYTESWAP(I);
305   1    5:1    17   FETCHWORD := I
306   1    5:0    17 END [ FETCHWORD ] ;
307   1    5:0    32
308   1    6:0     1 PROCEDURE STOREBYTE(VAL: INTEGER; CP: CODEP; OFFSET: INTEGER);
309   1    6:0     0 BEGIN
310   1    6:1     0   CP^[OFFSET] := VAL
311   1    6:0     2 END [ STOREBYTE ] ;
312   1    6:0    16
313   1    7:0     1 PROCEDURE STOREWORD(VAL: INTEGER; CP: CODEP; OFFSET: INTEGER);
314   1    7:0     0 BEGIN
315   1    7:0     0   [ BYTE SWAP VAL ]
316   1    7:1     0   IF FLIPPED THEN BYTESWAP(VAL);
317   1    7:1     9   MOVELEFT(VAL, CP^[OFFSET], 2)
318   1    7:0    17 END [ STOREWORD ] ;
319   1    7:0    30
320   1    8:0     1 PROCEDURE STOREBIG(VAL: INTEGER; CP: CODEP; OFFSET: INTEGER);
321   1    8:0     4   VAR  BIGWORD: PACKED RECORD
322   1    8:0     4                    CASE BOOLEAN OF
323   1    8:0     4                      TRUE:  (INTEG: INTEGER);
324   1    8:0     4                      FALSE: (LOWBYTE: 0..255;
325   1    8:0     4                              HIGHBYTE: 0..255)
326   1    8:0     4                  END;
327   1    8:0     0 BEGIN
```

```
328   1     8:1     0   BIGWORD.INTEG := VAL;
329   1     8:1     3   CP^[OFFSET] := BIGWORD.HIGHBYTE + 128;
330   1     8:1    15   CP^[OFFSET + 1] := BIGWORD.LOWBYTE;
331   1     8:0    25 END;
332   1     8:0    38
333   1     8:0    38 [$R+]
334   1     8:0    38
335   1     8:0    38 {
336   1     8:0    38 *  BYTE-FLIP WORD QUANTITIES IN SEGMENT DICTIONARY
337   1     8:0    38 *  FOR BYTE-FLIPPED FILE CASE ON READING AND WRITING
338   1     8:0    38 *  SEGTABLES.  CALLED BY PHASE1 AND PHASE3.
339   1     8:0    38 }
340   1     8:0    38
341   1     9:0     1 PROCEDURE FLIPTABLE(VAR TABLE: I5SEGTBL);
342   1     9:0     2   VAR  S: SEGRANGE;
343   1     9:0     3        WORD: RECORD
344   1     9:0     3                CASE BOOLEAN OF
345   1     9:0     3                  TRUE:  (INT: INTEGER);
346   1     9:0     3                  FALSE: (KIND: SEGKINDS)
347   1     9:0     3              END;
348   1     9:0     0 BEGIN
349   1     9:1     0   FOR S := 0 TO MAXSEG DO
350   1     9:2    17     WITH TABLE, DISKINFO[S] DO
351   1     9:3    29     BEGIN
352   1     9:4    29       BYTESWAP(CODEADDR);
353   1     9:4    32       BYTESWAP(CODELENG);
354   1     9:4    37       WORD.KIND := SEGKIND[S];
355   1     9:4    49       BYTESWAP(WORD.INT);
356   1     9:4    53       SEGKIND[S] := WORD.KIND;
357   1     9:3    64     END;
358   1     9:0    71 END;
359   1     9:0    86
360   1     9:0    86 {
361   1     9:0    86 *  ENTER NEWSYM IN SYMTAB TREE.  THE TREE IS BINARY FOR
362   1     9:0    86 *  DIFFERENT NAMES AND ENTRIES WITH THE SAME NAME ARE ENTERED
363   1     9:0    86 *  ONTO SIDEWAYS LINKS (SLINK).  NO CHECK IS MADE FOR DUP
364   1     9:0    86 *  ENTRY TYPES, CALLER MUST DO THAT.  NODES ON SLINK WILL
365   1     9:0    86 *  ALWAYS HAVE NIL RLINK AND LLINK.
366   1     9:0    86 }
367   1     9:0    86
368   1    10:0     1 PROCEDURE ENTERSYM(NEWSYM: SYMP; VAR SYMTAB: SYMP);
```

```
369   1   10:0    3   VAR SYP, LASTSYP: SYMP;
370   1   10:0    5       USELEFT: BOOLEAN;
371   1   10:0    0 BEGIN
372   1   10:1    0   NEWSYM^.LLINK := NIL;
373   1   10:1    5   NEWSYM^.RLINK := NIL;
374   1   10:1   10   NEWSYM^.SLINK := NIL;
375   1   10:1   13   IF SYMTAB = NIL THEN
376   1   10:2   19     SYMTAB := NEWSYM
377   1   10:1   20   ELSE
378   1   10:2   24     BEGIN [ SEARCH SYMTAB AND ADD NEWSYM ]
379   1   10:3   24       SYP := SYMTAB;
380   1   10:3   28       REPEAT
381   1   10:4   28         LASTSYP := SYP;
382   1   10:4   31         IF SYP^.ENTRY.NAME > NEWSYM^.ENTRY.NAME THEN
383   1   10:5   42           BEGIN SYP := SYP^.LLINK; USELEFT := TRUE END
384   1   10:4   49         ELSE
385   1   10:5   51           IF SYP^.ENTRY.NAME < NEWSYM^.ENTRY.NAME THEN
386   1   10:6   62             BEGIN SYP := SYP^.RLINK; USELEFT := FALSE END
387   1   10:5   69           ELSE [ EQUAL ]
388   1   10:6   71             BEGIN [ ADD INTO SIDEWAYS LIST ]
389   1   10:7   71               NEWSYM^.SLINK := SYP^.SLINK;
390   1   10:7   75               SYP^.SLINK := NEWSYM;
391   1   10:7   78               LASTSYP := NIL;      [ ALREADY ADDED FLAG ]
392   1   10:7   81               SYP := NIL           [ STOP REPEAT LOOP ]
393   1   10:6   81             END
394   1   10:3   84       UNTIL SYP = NIL;
395   1   10:3   89       IF LASTSYP <> NIL THEN
396   1   10:4   94         BEGIN [ ADD TO BOTTOM OF TREE ]
397   1   10:5   94           IF USELEFT THEN
398   1   10:6   97             LASTSYP^.LLINK := NEWSYM
399   1   10:5   00           ELSE
400   1   10:6   04             LASTSYP^.RLINK := NEWSYM
401   1   10:4   07         END
402   1   10:2   09     END [ SYMTAB <> NIL ]
403   1   10:0   09 END [ ENTERSYM ] ;
404   1   10:0   24
405   1   10:0   24 [
406   1   10:0   24 *  LOOK UP NAME IN SYMTAB TREE AND RETURN POINTER
407   1   10:0   24 *  TO IT.  OKTYPE RESTRICTS WHAT LITYPE IS
408   1   10:0   24 *  ACCEPTABLE.  NIL IS RETURNED IF NAME NOT FOUND.
409   1   10:0   24 ]
```

```
410   1   10:0    24
411   1   11:0     3 FUNCTION SYMSRCH(VAR NAME: ALPHA; OKTYPE: LITYPES; SYMTAB: SYMP): SYMP;
412   1   11:0     6   VAR SYP: SYMP;
413   1   11:0     0 BEGIN
414   1   11:1     0   SYMSRCH := NIL;
415   1   11:1     3   SYP := SYMTAB;
416   1   11:1     6   WHILE SYP <> NIL DO
417   1   11:2    11     IF SYP^.ENTRY.NAME > NAME THEN
418   1   11:3    20       SYP := SYP^.LLINK
419   1   11:2    21     ELSE
420   1   11:3    26       IF SYP^.ENTRY.NAME < NAME THEN
421   1   11:4    35         SYP := SYP^.RLINK
422   1   11:3    36       ELSE [ EQUAL NAME ]
423   1   11:4    41         IF SYP^.ENTRY.LITYPE <> OKTYPE THEN
424   1   11:5    47           SYP := SYP^.SLINK
425   1   11:4    48         ELSE [ FOUND! ]
426   1   11:5    53           BEGIN SYMSRCH := SYP; SYP := NIL END
427   1   11:0    59 END [ SYMSRCH ] ;
428   1   11:0    76
429   1   11:0    76 [
430   1   11:0    76 *  SEARCH FOR THE OCCURANCE OF THE UNIT SEGMENT
431   1   11:0    76 *  GIVEN BY NAME IN THE LIST OF FILES IN FP.
432   1   11:0    76 *  RETURN THE FILE AND SEGMENT NUMBER IN SEG.
433   1   11:0    76 *  NIL IS RETURNED FOR NON-EXISTANT UNITS AND
434   1   11:0    76 *  AN ERROR IS GIVEN.
435   1   11:0    76 ]
436   1   11:0    76
437   1   12:0     3 FUNCTION UNITSRCH(FP: FINFOP; VAR NAME: ALPHA; VAR SEG: SEGRANGE): FINFOP;
438   1   12:0     6   LABEL 1;
439   1   12:0     6   VAR S: SEGINDEX;
440   1   12:0     0 BEGIN SEG := 0;
441   1   12:1     6   WHILE FP <> NIL DO
442   1   12:2    11     BEGIN
443   1   12:3    11       WITH FP^.SEGTBL DO
444   1   12:4    16         FOR S := 0 TO MAXSEG DO
445   1   12:5    33           IF SEGNAME[S] = NAME THEN
446   1   12:6    48             IF SEGKIND[S] = UNITSEG THEN
447   1   12:7    62               GOTO 1;
448   1   12:3    71       FP := FP^.NEXT
449   1   12:2    72     END;
450   1   12:1    77   WRITE('UNIT ', NAME);
```

```
451   1   12:1    01   ERROR(' NOT FOUND');
452   1   12:1    16   S := 0;
453   1   12:1    22 1:
454   1   12:1    22   SEG := S;
455   1   12:1    28   UNITSRCH := FP
456   1   12:0    28 END [ UNITSRCH ] ;
457   1   12:0    48
458   1   12:0    48 [
459   1   12:0    48 *  ALPHABETIC RETURNS TRUE IF NAME CONTAINS ALL LEGAL
460   1   12:0    48 *  CHARACTERS FOR PASCAL IDENTIFIERS.  USED TO VALIDATE
461   1   12:0    48 *  SEGNAMES AND LINK INFO ENTRIES.
462   1   12:0    48 ]
463   1   12:0    48
464   1   13:0     3 FUNCTION ALPHABETIC(VAR NAME: ALPHA): BOOLEAN;
465   1   13:0     4   LABEL 1;
466   1   13:0     4   VAR I: INTEGER;
467   1   13:0     0 BEGIN
468   1   13:1     0   ALPHABETIC := FALSE;
469   1   13:1     3   FOR I := 0 TO 7 DO
470   1   13:2    14     IF NOT (NAME[I] IN ['A'..'Z', '0'..'9', ' ', '_']) THEN
471   1   13:3    39       GOTO 1;
472   1   13:1    48   ALPHABETIC := TRUE;
473   1   13:1    51 1:
474   1   13:0    51 END [ ALPHABETIC ] ;
475   1   13:0    66
476   1   13:0    66 [
477   1   13:0    66 *  GETCODEP IS A SNEAKY ROUTINE TO POINT CODEP'S ANYWHERE
478   1   13:0    66 *  IN MEMORY.  IT VIOLATES ROBOT'S RULES OF ORDER, BUT IS
479   1   13:0    66 *  VERY USEFUL FOR DEALING WITH THE VARIABLE SIZE SEGMENTS
480   1   13:0    66 ]
481   1   13:0    66
482   1   14:0     3 FUNCTION GETCODEP(MEMADDR: INTEGER): CODEP;
483   1   14:0     4   VAR R: RECORD
484   1   14:0     4             CASE BOOLEAN OF
485   1   14:0     4               TRUE:  (I: INTEGER);
486   1   14:0     4               FALSE: (P: CODEP)
487   1   14:0     4             END;
488   1   14:0     0 BEGIN
489   1   14:1     0   R.I := MEMADDR;
490   1   14:1     3   GETCODEP := R.P
491   1   14:0     3 END [ GETCODEP ] ;
```

```
492   1   14:0    18
493   1   14:0    18 [$I LINK0 ]
493   1   14:0    18 [$I LINK1 ]
494   1   14:0    18
495   1   14:0    18     (*************************************************************)
496   1   14:0    18     (*                                                           *)
497   1   14:0    18     (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.   *)
498   1   14:0    18     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-     *)
499   1   14:0    18     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE    *)
500   1   14:0    18     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.          *)
501   1   14:0    18     (*                                                           *)
502   1   14:0    18     (*************************************************************)
503   1   14:0    18
504   1   14:0    18 [
505   1   14:0    18 *  PHASE 1 OPENS HOST AND LIBRARY FILES AND
506   1   14:0    18 *  READS IN SEG TABLES.  ALL FIELDS ARE VERIFIED
507   1   14:0    18 *  AND THE HOSTFILE/LIBFILES FILE LIST IS BUILT.
508   1   14:0    18 *  THE PROTOTYPE FINAL SEG TABLE IS SET UP IN
509   1   14:0    18 *  SEGINFO[*] FROM THE HOST FILE AND THE SEP SEG
510   1   14:0    18 *  LIST IS SET UP FOR SEARCHING IN LATER PHASES.
511   1   14:0    18 ]
512   1   14:0    18
513   1   15:0     1 PROCEDURE PHASE1;
514   1   15:0     1
515   1   15:0     1   VAR  [ FOR USE WITH BYTE FLIPPING ]
516   1   15:0     1          HIGHBYTE: 0..1;
517   1   15:0     2          INT: RECORD
518   1   15:0     2                 CASE BOOLEAN OF
519   1   15:0     2                   TRUE:  (VAL: INTEGER);
520   1   15:0     2                   FALSE: (BYTE: PACKED ARRAY [0..1] OF 0..255)
521   1   15:0     2               END;
522   1   15:0     3     [
523   1   15:0     3     *  BUILD FILE LIST OPENS INPUT CODE FILES AND READS SEGTBLS.
524   1   15:0     3     *  THE VAR HOSTFILE IS SET UP AS HEAD OF LINKED LIST OF FILE
525   1   15:0     3     *  INFO RECS.  THE ORDER OF THESE FILES DETERMINES HOW ID'S
526   1   15:0     3     *  WILL BE SEARCHED FOR.  NOTE THAT LIBFILES POINTS AT THE
527   1   15:0     3     *  LIST JUST PAST THE HOST FILE FRONT ENTRY.
528   1   15:0     3     ]
529   1   15:0     3
530   1   16:0     1     PROCEDURE BUILDFILELIST;
531   1   16:0     1       LABEL 1;
```

```
532   1   16:0    1        VAR F: 0..MAXFILE;
533   1   16:0    2            I: INTEGER;
534   1   16:0    3            P, Q: FINFOP;
535   1   16:0    5            FNAME: STRING[39];
536   1   16:0   25
537   1   16:0   25          [
538   1   16:0   25          *  SETUPFILE OPENS FILE AND ENTERS NEW FINFO REC IN
539   1   16:0   25          *  HOSTFILE LIST.  SEGTBL IS READ IN AND VALIDATED.
540   1   16:0   25          ]
541   1   16:0   25
542   1   17:0    1          PROCEDURE SETUPFILE(NUM: INTEGER; KIND: FILEKIND; TITLE: STRING);
543   1   17:0   45            LABEL 1;
544   1   17:0   45            VAR ERRS: INTEGER;
545   1   17:0   46                S: SEGINDEX;
546   1   17:0   47                CP: FILEP;
547   1   17:0   48                FP: FINFOP;
548   1   17:0   49                ALLLINKED: BOOLEAN;
549   1   17:0   50                GOODKINDS: SET OF SEGKINDS;
550   1   17:0   51
551   1   17:0   51              [
552   1   17:0   51              *  GETFILEP RETURNS A POINTER TO A FILE USING UNSPEAKABLE
553   1   17:0   51              *  METHODS, BUT THE ENDS JUSTIFY THE MEANS.
554   1   17:0   51              ]
555   1   17:0   51
556   1   18:0    3             FUNCTION GETFILEP(VAR F: CODEFILE): FILEP;
557   1   18:0    4               VAR A: ARRAY [0..0] OF FILEP;
558   1   18:0    0             BEGIN
559   1   18:0    0               [$R-]
560   1   18:1    0               GETFILEP := A[-1];
561   1   18:1    9               [$R+]
562   1   18:0    9             END [ GETFILEP ] ;
563   1   18:0   22
564   1   17:0    0           BEGIN [ SETUPFILE ]
565   1   17:1    0             CASE NUM OF
566   1   17:1    8               0:  CP := GETFILEP(F0);
567   1   17:1   19               1:  CP := GETFILEP(F1);
568   1   17:1   30               2:  CP := GETFILEP(F2);
569   1   17:1   41               3:  CP := GETFILEP(F3);
570   1   17:1   52               4:  CP := GETFILEP(F4);
571   1   17:1   63               5:  CP := GETFILEP(F5);
572   1   17:1   74               6:  CP := GETFILEP(F6);
```

```
573   1   17:1    85            7:  CP := GETFILEP(F7)
574   1   17:1    87          END [ CASES ] ;
575   1   17:1    18          RESET(CP^, TITLE);
576   1   17:1    27          IF IORESULT <> 0 THEN
577   1   17:2    33            IF TITLE <> 'IN WORKSPACE' THEN
578   1   17:3    54              BEGIN
579   1   17:4    54                INSERT('.CODE', TITLE, LENGTH(TITLE)+1);
580   1   17:4    74                RESET(CP^, TITLE)
581   1   17:3    83              END;
582   1   17:1    83          IF IORESULT <> 0 THEN
583   1   17:2    89            BEGIN
584   1   17:3    89              INSERT('NO FILE ', TITLE, 1);
585   1   17:3    07              ERROR(TITLE);
586   1   17:3    11              IF KIND <> USERHOST THEN
587   1   17:4    16                ERRCOUNT := ERRCOUNT-1
588   1   17:2    17            END
589   1   17:1    21          ELSE
590   1   17:2    23            BEGIN [ FILE OPEN OK ]
591   1   17:3    23              IF TALKATIVE THEN
592   1   17:4    26                WRITELN('OPENING ', TITLE);
593   1   17:3    59              NEW(FP);
594   1   17:3    66              FP^.NEXT := HOSTFILE;
595   1   17:3    70              FP^.CODE := CP;
596   1   17:3    77              FP^.FKIND := KIND;
597   1   17:3    83              IF BLOCKREAD(CP^, FP^.SEGTBL, 1, 0) <> 1 THEN
598   1   17:4    02                ERROR('SEGTBL READ ERR')
599   1   17:3    20              ELSE
600   1   17:4    24                BEGIN [ NOW CHECK SEGTBL VALUES ]
601   1   17:5    24                  IF NUM = 0 THEN [ DETERMINE IF FILE IS BYTE-FLIPPED ]
602   1   17:6    29                    FOR S := 0 TO MAXSEG DO
603   1   17:7    48                    BEGIN
604   1   17:8    48                      INT.VAL := ORD(FP^.SEGTBL.SEGKIND[S]);
605   1   17:8    63                      FLIPPED := (INT.BYTE[HIGHBYTE] <> 0);
606   1   17:8    78                      IF FLIPPED THEN
607   1   17:9    83                        GOTO 1;
608   1   17:7    85                    END;
609   1   17:5    93                1:  IF FLIPPED THEN
610   1   17:6    98                      FLIPTABLE(FP^.SEGTBL);
611   1   17:6    04
612   1   17:5    04                  S := 0; ALLLINKED := TRUE;
613   1   17:5    13                  ERRS := ERRCOUNT;
```

```
614   1   17:5    16                IF KIND = USERHOST THEN
615   1   17:6    21                  GOODKINDS := [LINKED,SEGPROC,SEPRTSEG,HOSTSEG,UNITSEG]
616   1   17:5    21                ELSE
617   1   17:6    29                  GOODKINDS := [LINKED,UNITSEG,SEPRTSEG];
618   1   17:6    35
619   1   17:5    35                WITH FP^.SEGTBL DO
620   1   17:6    41                  REPEAT
621   1   17:7    41                    INT.VAL := ORD(SEGKIND[S]);
622   1   17:7    56                    IF (INT.BYTE[HIGHBYTE] <> 0) THEN
623   1   17:8    70                    BEGIN
624   1   17:9    70                      ERROR('BAD BYTE SEX'); EXIT(LINKER)
625   1   17:8    91                    END;
626   1   17:7    91                    ALLLINKED := ALLLINKED AND (SEGKIND[S] = LINKED);
627   1   17:7    10                    IF (DISKINFO[S].CODELENG = 0)
628   1   17:7    22                    AND (SEGKIND[S] <> LINKED) THEN
629   1   17:8    39                      IF (KIND <> USERHOST)
630   1   17:8    42                      OR (SEGKIND[S] <> UNITSEG) THEN
631   1   17:9    59                        ERROR('FUNNY CODE SEG');
632   1   17:7    78                    IF (DISKINFO[S].CODELENG < 0)
633   1   17:7    90                    OR (DISKINFO[S].CODEADDR < 0)
634   1   17:7    02                    OR (DISKINFO[S].CODEADDR > 300) THEN
635   1   17:8    20                      ERROR('BAD DISKINFO');
636   1   17:7    37                    IF NOT (SEGKIND[S] IN GOODKINDS) THEN
637   1   17:8    56                      ERROR('BAD SEG KIND');
638   1   17:7    73                    IF NOT ALPHABETIC(SEGNAME[S]) THEN
639   1   17:8    91                      ERROR('BAD SEG NAME');
640   1   17:7    08                    IF ERRCOUNT > ERRS THEN
641   1   17:8    14                      S := MAXSEG;
642   1   17:7    20                    S := S+1
643   1   17:6    22                  UNTIL S > MAXSEG;
644   1   17:5    35                IF ALLLINKED AND (KIND = USERHOST) THEN
645   1   17:6    43                  BEGIN
646   1   17:7    43                    WRITE('ALL SEGS LINKED');
647   1   17:7    68                    EXIT(LINKER)
648   1   17:6    72                  END;
649   1   17:5    72                IF ERRCOUNT = ERRS THEN
650   1   17:6    78                  HOSTFILE := FP              [ OK FILE...LINK IN ]
651   1   17:4    78              END
652   1   17:2    82          END
653   1   17:0    82      END [ SETUPFILE ] ;
654   1   17:0    02
```

```
655   1   16:0     0    BEGIN { BUILDFILELIST }
656   1   16:1     0      IF TALKATIVE THEN
657   1   16:2     3        BEGIN
658   1   16:3     3          FOR I := 1 TO 7 DO
659   1   16:4    15            WRITELN;
660   1   16:3    28          WRITELN( HEADER )
661   1   16:2    57        END;
662   1   16:1    57      USEWORKFILE := CMDSTATE <> SYSPROG;
663   1   16:1    64      WITH USERINFO DO
664   1   16:2    64        IF USEWORKFILE THEN
665   1   16:3    67          BEGIN
666   1   16:4    67            IF GOTCODE THEN
667   1   16:5    72              FNAME := CONCAT(CODEVID, ':', CODETID)
668   1   16:4    07            ELSE
669   1   16:5    11              FNAME := 'IN WORKSPACE';
670   1   16:4    30            SETUPFILE(0, USERHOST, FNAME);
671   1   16:4    36            SETUPFILE(1, SYSTEMLIB, '*SYSTEM.LIBRARY')
672   1   16:3    56          END
673   1   16:2    58        ELSE
674   1   16:3    60          BEGIN
675   1   16:4    60            WRITE('HOST FILE? ');
676   1   16:4    81            READLN(FNAME);
677   1   16:4    96            IF FNAME = '' THEN
678   1   16:5    05              IF GOTCODE THEN
679   1   16:6    10                FNAME := CONCAT(CODEVID, ':', CODETID)
680   1   16:5    45              ELSE
681   1   16:6    49                FNAME := 'IN WORKSPACE';
682   1   16:4    68            SETUPFILE(0, USERHOST, FNAME);
683   1   16:4    74            IF ERRCOUNT > 0 THEN
684   1   16:5    79              EXIT(LINKER); { NO HOST! }
685   1   16:4    83            FOR F := 1 TO MAXFILE DO
686   1   16:5    01              BEGIN
687   1   16:6    01                WRITE('LIB FILE? ');
688   1   16:6    21                READLN(FNAME);
689   1   16:6    36                IF FNAME = '' THEN
690   1   16:7    45                  GOTO 1;
691   1   16:6    47                IF FNAME = '*' THEN
692   1   16:7    54                  SETUPFILE(F, SYSTEMLIB, '*SYSTEM.LIBRARY')
693   1   16:6    74                ELSE
694   1   16:7    78                  SETUPFILE(F, USERLIB, FNAME)
695   1   16:5    82              END;
```

```
696   1   16:4    91          1:
697   1   16:4    91              WRITE('MAP NAME? ');
698   1   16:4    11              READLN(MAPNAME);
699   1   16:4    26              IF MAPNAME <> '' THEN
700   1   16:5    35                IF MAPNAME[LENGTH(MAPNAME)] = '.' THEN
701   1   16:6    47                  DELETE(MAPNAME, LENGTH(MAPNAME), 1)
702   1   16:5    57                ELSE
703   1   16:6    59                  INSERT('.TEXT', MAPNAME, LENGTH(MAPNAME)+1)
704   1   16:3    79            END;
705   1   16:3    79
706   1   16:3    79        [ NOW REVERSE LIST SO HOST IS ]
707   1   16:3    79        [ FIRST AND SYSLIB IS LAST    ]
708   1   16:3    79
709   1   16:1    79        P := HOSTFILE; HOSTFILE := NIL;
710   1   16:1    85        REPEAT
711   1   16:2    85          Q := P^.NEXT;
712   1   16:2    89          P^.NEXT := HOSTFILE;
713   1   16:2    92          HOSTFILE := P;
714   1   16:2    95          P := Q
715   1   16:1    95        UNTIL P = NIL;
716   1   16:1    03        LIBFILES := HOSTFILE^.NEXT;
717   1   16:0    07      END [ BUILDFILELIST ] ;
718   1   16:0    28
719   1   16:0    28      [
720   1   16:0    28      *  BUILDSEGINFO INITIALIZES THE SEGINFO TABLE FROM
721   1   16:0    28      *  THE HOST PROTOTYPE SEG TABLE.  ALL LEGAL STATES
722   1   16:0    28      *  ARE CHECKED, AND IMPORTED UNITS FOUND.  THIS
723   1   16:0    28      *  LEAVES A LIST OF ALL SEGS TO FINALLY APPEAR IN
724   1   16:0    28      *  THE OUTPUT CODE FILE.
725   1   16:0    28      ]
726   1   16:0    28
727   1   19:D     1      PROCEDURE BUILDSEGINFO;
728   1   19:D     1        LABEL 1;
729   1   19:D     1        VAR S: SEGINDEX;
730   1   19:D     2            ERRS: INTEGER;
731   1   19:D     3            SP: SEGP;
732   1   19:0     0      BEGIN
733   1   19:1     0        WITH HOSTFILE^.SEGTBL DO
734   1   19:2     5          FOR S := 0 TO MAXSEG DO
735   1   19:3    22            IF (SEGKIND[S] = LINKED)
736   1   19:3    34            AND (DISKINFO[S].CODELENG = 0) THEN
```

```
737   1    19:4    47          SEGINFO[S] := NIL   [ NOT IN USE ]
738   1    19:3    55        ELSE
739   1    19:4    59          BEGIN [ DO SOMETHING WITH SEG ]
740   1    19:5    59            ERRS := ERRCOUNT;
741   1    19:5    62            NEW(SP);
742   1    19:5    67            SP^.SRCFILE := HOSTFILE;
743   1    19:5    70            SP^.SRCSEG := S;
744   1    19:5    78            SP^.SYMTAB := NIL;
745   1    19:5    83            SP^.SEGKIND := SEGKIND[S];
746   1    19:5    97            CASE SP^.SEGKIND OF
747   1    19:5    01              SEGPROC,
748   1    19:5    01              LINKED:     ;  [ NOTHING TO CHECK! ]
749   1    19:5    03
750   1    19:5    03              HOSTSEG:    IF S <> MASTERSEG THEN
751   1    19:7    08                            ERROR('BAD HOST SEG')
752   1    19:6    23                          ELSE
753   1    19:7    27                            IF HOSTSP <> NIL THEN
754   1    19:8    32                              ERROR('DUP HOST SEG')
755   1    19:7    47                            ELSE
756   1    19:8    51                              HOSTSP := SP;
757   1    19:8    56
758   1    19:5    56              SEPRTSEG:   IF S = MASTERSEG THEN
759   1    19:7    61                            SP^.NEXT := NIL
760   1    19:6    64                          ELSE
761   1    19:7    68                            BEGIN [ PUT INTO SEPLIST ]
762   1    19:8    68                              SP^.NEXT := SEPLIST;
763   1    19:8    73                              SEPLIST := SP;
764   1    19:8    76                              SP := NIL
765   1    19:7    76                            END;
766   1    19:7    81
767   1    19:5    81              UNITSEG:    IF DISKINFO[S].CODELENG = 0 THEN
768   1    19:7    93                            SP^.SRCFILE := UNITSRCH(LIBFILES,
769   1    19:7    95                                                    SEGNAME[S],
770   1    19:7    04                                                    SP^.SRCSEG)
771   1    19:5    07            END [ CASES ] ;
772   1    19:5    32            IF ERRS = ERRCOUNT THEN
773   1    19:6    37              SEGINFO[S] := SP
774   1    19:5    45            ELSE
775   1    19:6    49              SEGINFO[S] := NIL
776   1    19:4    57          END;
777   1    19:4    66
```

```
778   1   19:4    66       [ NOW FIND FIRST ASSIGNABLE SEG ]
779   1   19:4    66
780   1   19:1    66       FOR S := FIRSTSEG TO MAXSEG DO
781   1   19:2    83         IF SEGINFO[S] = NIL THEN
782   1   19:3    96           GOTO 1;
783   1   19:1    05       S := MAXSEG1;
784   1   19:1    11     1:
785   1   19:1    11       NEXTSEG := S;
786   1   19:1    17       IF SEGINFO[MASTERSEG] = NIL THEN
787   1   19:2    30         ERROR('WEIRD HOST')
788   1   19:0    43     END [ BUILDSEGINFO ] ;
789   1   19:0    68
790   1   19:0    68     [
791   1   19:0    68     *  BUILDSEPLIST SEARCHES THROUGH LIBRARIES AND ADDS ONTO
792   1   19:0    68     *  A GLOBAL LIST OF SEP SEGS THAT ARE TO BE SEARCHED
793   1   19:0    68     *  FOR PROCS AND GLOBALS.  THEY ARE INITIALLY BUILT IN
794   1   19:0    68     *  THE REVERSE ORDER, THEN REVERSED AGAIN SO SEARCHES
795   1   19:0    68     *  WILL GO IN THE ORDER THE FILES WERE SPECIFIED.
796   1   19:0    68     ]
797   1   19:0    68
798   1   20:D     1     PROCEDURE BUILDSEPLIST;
799   1   20:D     1       VAR SP, P, Q: SEGP;
800   1   20:D     4           FP: FINFOP;
801   1   20:D     5           S: SEGINDEX;
802   1   20:0     0     BEGIN
803   1   20:1     0       FP := LIBFILES;
804   1   20:1     3       WHILE FP <> NIL DO
805   1   20:2     8         BEGIN
806   1   20:3     8           FOR S := 0 TO MAXSEG DO
807   1   20:4    25             IF FP^.SEGTBL.SEGKIND[S] = SEPRTSEG THEN
808   1   20:5    39               BEGIN
809   1   20:6    39                 NEW(SP);
810   1   20:6    44                 SP^.NEXT := SEPLIST;
811   1   20:6    49                 SP^.SRCFILE := FP;
812   1   20:6    52                 SP^.SRCSEG := S;
813   1   20:6    60                 SP^.SYMTAB := NIL;
814   1   20:6    65                 SP^.SEGKIND := SEPRTSEG;
815   1   20:6    70                 SP^.NEXT := SEPLIST;
816   1   20:6    75                 SEPLIST := SP
817   1   20:5    75               END;
818   1   20:3    85           FP := FP^.NEXT
```

```
819   1   20:2    86          END;
820   1   20:2    91
821   1   20:2    91        [ NOW REVERSE THE LIST TO MAINTAIN ORIGINAL ORDER ]
822   1   20:2    91
823   1   20:1    91        P := SEPLIST; SEPLIST := NIL;
824   1   20:1    97        WHILE P <> NIL DO
825   1   20:2    02          BEGIN
826   1   20:3    02            Q := P^.NEXT;
827   1   20:3    06            P^.NEXT := SEPLIST;
828   1   20:3    11            SEPLIST := P;
829   1   20:3    14            P := Q
830   1   20:2    14          END
831   1   20:0    17      END [ BUILDSEPLIST ] ;
832   1   20:0    38
833   1   15:0     0  BEGIN [ PHASE1 ]
834   1   15:0     0
835   1   15:0     0    [ INITIALIZE GLOBALS ]
836   1   15:0     0
837   1   15:1     0    HOSTFILE := NIL;
838   1   15:1     3    LIBFILES := NIL;
839   1   15:1     6    HOSTSP := NIL;
840   1   15:1     9    SEPLIST := NIL;
841   1   15:1    12    REFLITYPES := [UNITREF, GLOBREF, PUBLREF,
842   1   15:1    12                   PRIVREF, CONSTREF,
843   1   15:1    12                   SEPPREF, SEPFREF];
844   1   15:1    20    ERRCOUNT := 0;
845   1   15:1    23    NEXTBASELC := 3;
846   1   15:1    31    MAPNAME := '';
847   1   15:1    38    TALKATIVE := NOT USERINFO.SLOWTERM;
848   1   15:1    44    MARK(HEAPBASE);
849   1   15:1    48    UNITWRITE(3, HEAPBASE^, 35);
850   1   15:1    56
851   1   15:1    56    [ DETERMINE BYTE SEX OF MACHINE ]
852   1   15:1    56
853   1   15:1    56    FLIPPED := FALSE;
854   1   15:1    60    INT.VAL := 1;
855   1   15:1    63    HIGHBYTE := ORD( INT.BYTE[0] = 1 );
856   1   15:1    77
857   1   15:1    77    [ BUILD LIST OF INPUT FILES ]
858   1   15:1    77
859   1   15:1    77    BUILDFILELIST;
```

```
860   1   15:1    79   IF ERRCOUNT > 0 THEN
861   1   15:2    84     EXIT(LINKER);
862   1   15:2    88
863   1   15:2    88   [ INIT BASIC SEG INFO TABLE ]
864   1   15:2    88
865   1   15:1    88   BUILDSEGINFO;
866   1   15:1    90   IF ERRCOUNT > 0 THEN
867   1   15:2    95     EXIT(LINKER);
868   1   15:2    99
869   1   15:2    99   [ FINALLY BUILD SEP SEG LIST ]
870   1   15:2    99
871   1   15:1    99   BUILDSEPLIST;
872   1   15:1    01   IF ERRCOUNT > 0 THEN
873   1   15:2    06     EXIT(LINKER)
874   1   15:0    10 END [ PHASE1 ] ;
875   1   15:0    22
876   1   15:0    22 [$I LINK1 ]
876   1   15:0    22 [$I LINK2 ]
877   1   15:0    22
878   1   15:0    22     (*****************************************************************)
879   1   15:0    22     (*                                                               *)
880   1   15:0    22     (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
881   1   15:0    22     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
882   1   15:0    22     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
883   1   15:0    22     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
884   1   15:0    22     (*                                                               *)
885   1   15:0    22     (*****************************************************************)
886   1   15:0    22
887   1   15:0    22 [
888   1   15:0    22 *  PHASE2 READS IN ALL LINKER INFO ASSOCIATED WITH
889   1   15:0    22 *  THE SEGS IN SEGINFO AND SEP SEG LIST.  AGAIN ALL
890   1   15:0    22 *  FIELDS ARE CHECKED CAREFULLY.  AS A HELP TO PHASE3,
891   1   15:0    22 *  REF LISTS ARE COLLECTED AND PLACE RECORDS FOR SEP
892   1   15:0    22 *  PROC/FUNC ARE COMPUTED.  SOME SMALL OPTIMIZATION IS
893   1   15:0    22 *  DONE TO ELIMINATE THE SEP SEG LIST IF IT IS NOT
894   1   15:0    22 *  GOING TO BE NEEDED, SAVING A FEW DISK IO'S.
895   1   15:0    22 ]
896   1   15:0    22
897   1   21:D     1 PROCEDURE PHASE2;
898   1   21:D     1   VAR S: SEGINDEX;
899   1   21:D     2       SP: SEGP;
```

```
900   1   21:0    3       DUMPSEPS: BOOLEAN;
901   1   21:0    4
902   1   21:0    4     [
903   1   21:0    4     *  READLINKINFO READS IN THE LINK INFO FOR SEGMENT SP
904   1   21:0    4     *  AND BUILDS ITS SYMTAB.  SOME SIMPLE DISK IO ROUTINES
905   1   21:0    4     *  DO UNBLOCKING, AND ALL FIELDS ARE AGAIN VERIFIED.
906   1   21:0    4     *  THE ONLY LEGAL LITYPES ARE IN OKTYPES. ASSUME THAT
907   1   21:0    4     *  SP <> NIL
908   1   21:0    4     ]
909   1   21:0    4
910   1   22:0    1     PROCEDURE READLINKINFO(SP: SEGP; OKTYPES: LISET);
911   1   22:0    3       VAR RP, RQ: REFP;
912   1   22:0    5           SYP: SYMP;
913   1   22:0    6           W, ERRS, NRECS, NEXTBLK, RECSLEFT: INTEGER;
914   1   22:0   11           ENTRY, TEMP: LIENTRY;
915   1   22:0   29           BUF: ARRAY [0..31] OF
916   1   22:0   29                  ARRAY [0..7] OF INTEGER;
917   1   22:0   85           TENTRY: ARRAY [0..7] OF INTEGER;
918   1   22:0   93
919   1   22:0   93         [
920   1   22:0   93         *  GETENTRY READS AN 8 WORD RECORD FROM DISK BUF
921   1   22:0   93         *  SEQUENTIALLY.  NO VALIDITY CHECKING IS DONE HERE,
922   1   22:0   93         *  ONLY DISK READ ERRORS.
923   1   22:0   93         ]
924   1   22:0   93
925   1   23:0    1         PROCEDURE GETENTRY(VAR ENTRY: LIENTRY);
926   1   23:0    2           VAR ERR: BOOLEAN;
927   1   23:0    0         BEGIN
928   1   23:1    0           ERR := FALSE;
929   1   23:1    3           IF RECSLEFT = 0 THEN
930   1   23:2   10             BEGIN
931   1   23:3   10               RECSLEFT := 32;
932   1   23:3   14               ERR := BLOCKREAD(SP^.SRCFILE^.CODE^, BUF, 1, NEXTBLK) <> 1;
933   1   23:3   37               IF ERR THEN
934   1   23:4   40                 ERROR('LI READ ERR')
935   1   23:3   54               ELSE
936   1   23:4   58                 NEXTBLK := NEXTBLK+1
937   1   23:2   61             END;
938   1   23:1   66           MOVELEFT(BUF[32-RECSLEFT], ENTRY, 16);
939   1   23:1   85           IF ERR THEN
940   1   23:2   88             ENTRY.LITYPE := EOFMARK;
```

```
941   1   23:1    93          RECSLEFT := RECSLEFT-1
942   1   23:0    96        END [ GETENTRY ] ;
943   1   23:0    14
944   1   23:0    14        [
945   1   23:0    14        *  ADDUNIT IS CALLED TO FIND OR ALLOCATE A LIBRARY UNIT
946   1   23:0    14        *  THAT IS FOUND IN LINK INFO AS AN EXTERNAL REF.  THIS
947   1   23:0    14        *  OCCURS IN LIB UNITS WHICH USE OTHER UNITS.  IF
948   1   23:0    14        *  THE UNIT CAN'T BE FOUND OR NO ROOM, ERROR IS CALLED.
949   1   23:0    14        ]
950   1   23:0    14
951   1   24:0     1        PROCEDURE ADDUNIT(VAR NAME: ALPHA);
952   1   24:0     2          VAR FP: FINFOP; SEG: INTEGER;
953   1   24:0     0        BEGIN
954   1   24:1     0          FP := UNITSRCH(HOSTFILE, NAME, SEG);
955   1   24:1    10          IF FP <> NIL THEN
956   1   24:2    15            IF FP <> HOSTFILE THEN
957   1   24:3    20              IF FP^.SEGTBL.DISKINFO[SEG].CODELENG <> 0 THEN
958   1   24:4    34                IF NEXTSEG = MAXSEG1 THEN
959   1   24:5    40                  ERROR('NO ROOM IN SEGINFO')
960   1   24:4    61                ELSE
961   1   24:5    65                  BEGIN [ ALLOCATE NEW SEGINFO EL ]
962   1   24:6    65                    NEW(SEGINFO[NEXTSEG]);
963   1   24:6    77                    WITH SEGINFO[NEXTSEG]^ DO
964   1   24:7    89                      BEGIN
965   1   24:8    89                        SRCFILE := FP;
966   1   24:8    92                        SRCSEG := SEG;
967   1   24:8    00                        SEGKIND := UNITSEG;
968   1   24:8    05                        SYMTAB := NIL
969   1   24:7    08                      END;
970   1   24:6    10                    NEXTSEG := NEXTSEG+1
971   1   24:5    12                  END
972   1   24:0    19        END [ ADDUNIT ] ;
973   1   24:0    32
974   1   24:0    32        [
975   1   24:0    32        *  VALIDATE VERIFIES LIENTRY FORMAT.
976   1   24:0    32        *  IF THE ENTRY IS SEPPROC OR FUNC
977   1   24:0    32        *  THEN A PLACE REC IS ALLOCATED FOR BUILDPLACE.  IF
978   1   24:0    32        *  A UNITREF IS FOUND, IT SEARCHED FOR AND POSSIBLY
979   1   24:0    32        *  ALLOCATED.  IF THE UNIT MUST BE ADDED TO SEGINFO,
980   1   24:0    32        *  IT IS PLACED AFTER CURRENT POSITION SO IT WILL HAVE
981   1   24:0    32        *  ITS LINK INFO READ AS WELL.
```

```
 982   1   24:0    32          ]
 983   1   24:0    32
 984   1   25:0     1          PROCEDURE VALIDATE(VAR ENTRY: LIENTRY);
 985   1   25:0     0          BEGIN
 986   1   25:1     0            WITH ENTRY DO
 987   1   25:2     3              IF NOT ALPHABETIC(NAME) THEN
 988   1   25:3    11                ERROR('NON-ALPHA NAME')
 989   1   25:2    28              ELSE
 990   1   25:3    32                CASE LITYPE OF
 991   1   25:3    36                  SEPPREF,
 992   1   25:3    36                  SEPFREF,
 993   1   25:3    36                  UNITREF,
 994   1   25:3    36                  GLOBREF,
 995   1   25:3    36                  PUBLREF,
 996   1   25:3    36                  PRIVREF,
 997   1   25:3    36                  CONSTREF:  BEGIN
 998   1   25:5    36                               REFLIST := NIL;
 999   1   25:5    41                               IF (NREFS < 0)
1000   1   25:5    45                               OR (NREFS > 500) THEN
1001   1   25:6    54                                 ERROR('TOO MANY REFS');
1002   1   25:5    72                               IF NOT (FORMAT IN [WORD, BYTE, BIG]) THEN
1003   1   25:6    80                                 ERROR('BAD FORMAT');
1004   1   25:5    95                               IF LITYPE = PRIVREF THEN
1005   1   25:6    01                                 IF (NWORDS <= 0)
1006   1   25:6    05                                 OR (NWORDS > MAXLC) THEN
1007   1   25:7    14                                   ERROR('BAD PRIVATE');
1008   1   25:5    30                               IF LITYPE = UNITREF THEN
1009   1   25:6    36                                 IF NREFS <> 0 THEN
1010   1   25:7    42                                   ADDUNIT(NAME)
1011   1   25:4    43                             END;
1012   1   25:3    47                  GLOBDEF:   IF (HOMEPROC <= 0)
1013   1   25:4    51                             OR (HOMEPROC > MAXPROC)
1014   1   25:4    57                             OR (ICOFFSET < 0)
1015   1   25:4    62                             OR (ICOFFSET > MAXIC) THEN
1016   1   25:5    72                               ERROR('BAD GLOBDEF');
1017   1   25:3    90                  PUBLDEF:   IF (BASEOFFSET <= 0)
1018   1   25:4    94                             OR (BASEOFFSET > MAXLC) THEN
1019   1   25:5    03                               ERROR('BAD PUBLICDEF');
1020   1   25:3    23                  EXTPROC,
1021   1   25:3    23                  EXTFUNC,
1022   1   25:3    23                  SEPPROC,
```

```
1023   1   25:3    23                SEPFUNC:   BEGIN
1024   1   25:5    23                             IF LITYPE IN [SEPPROC,SEPFUNC] THEN
1025   1   25:6    32                               NEW(PLACE) [ FOR USE IN BUILDPLACES ]
1026   1   25:5    38                             ELSE
1027   1   25:6    40                               PLACE := NIL;
1028   1   25:5    45                             IF (SRCPROC <= 0)
1029   1   25:5    49                             OR (SRCPROC > MAXPROC)
1030   1   25:5    55                             OR (NPARAMS < 0)
1031   1   25:5    60                             OR (NPARAMS > 100) THEN
1032   1   25:6    68                               ERROR('BAD PROC/FUNC')
1033   1   25:4    84                           END
1034   1   25:3    86              END [ CASE LITYPE ]
1035   1   25:0    24        END [ VALIDATE ] ;
1036   1   25:0    42
1037   1   22:0     0      BEGIN [ READLINKINFO ]
1038   1   22:1     0        RECSLEFT := 0;       [ 8 WD RECS LEFT IN BUF ]
1039   1   22:1     3        WITH SP^.SRCFILE^.SEGTBL, DISKINFO[SP^.SRCSEG] DO
1040   1   22:2    23          BEGIN [ SEEK TO LINKINFO ]
1041   1   22:3    23            NEXTBLK := CODEADDR + (CODELENG+511) DIV 512;
1042   1   22:3    42            IF TALKATIVE THEN
1043   1   22:4    45              WRITELN('READING ', SEGNAME[SP^.SRCSEG])
1044   1   22:2    89          END;
1045   1   22:1    89        REPEAT
1046   1   22:2    89          GETENTRY(ENTRY);
1047   1   22:2    93          IF FLIPPED THEN [ FLIP WORD QUANTITIES IN LIENTRY ]
1048   1   22:3    98          BEGIN
1049   1   22:4    98            MOVELEFT(ENTRY, TENTRY, 16);
1050   1   22:4    08            FOR W := 4 TO 7 DO
1051   1   22:5    22              BYTESWAP(TENTRY[W]);
1052   1   22:4    40            MOVELEFT(TENTRY, ENTRY, 16);
1053   1   22:3    50          END;
1054   1   22:2    50          ERRS := ERRCOUNT;
1055   1   22:2    53          IF ENTRY.LITYPE <> EOFMARK THEN
1056   1   22:3    59            IF ENTRY.LITYPE IN OKTYPES THEN
1057   1   22:4    66              VALIDATE(ENTRY)
1058   1   22:3    68            ELSE
1059   1   22:4    72              BEGIN
1060   1   22:5    72                ERROR('BAD LITYPE');
1061   1   22:5    87                ENTRY.LITYPE := EOFMARK
1062   1   22:4    87              END;
1063   1   22:2    90          IF DUMPSEPS THEN
```

```
1064   1   22:3    95            IF ENTRY.LITYPE IN [SEPPREF, SEPFREF,
1065   1   22:3    97                                EXTPROC, EXTFUNC,
1066   1   22:3    97                                GLOBREF] THEN
1067   1   22:4    04              DUMPSEPS := FALSE;  { WE NEED THEM! }
1068   1   22:2    08          IF ENTRY.LITYPE IN REFLITYPES THEN
1069   1   22:3    15            BEGIN { READ REF LIST }
1070   1   22:4    15              NRECS := (ENTRY.NREFS+7) DIV 8;
1071   1   22:4    23              WHILE NRECS > 0 DO
1072   1   22:5    28                BEGIN { READ REF REC }
1073   1   22:6    28                  GETENTRY(TEMP);
1074   1   22:6    32                  NEW(RP);
1075   1   22:6    37                  MOVELEFT(TEMP, RP^.REFS, 16);
1076   1   22:6    47                  IF FLIPPED THEN { FLIP REF WORDS }
1077   1   22:7    52                    FOR W := 0 TO 7 DO
1078   1   22:8    66                      BYTESWAP(RP^.REFS[W]);
1079   1   22:6    84                  RP^.NEXT := ENTRY.REFLIST;
1080   1   22:6    88                  ENTRY.REFLIST := RP;
1081   1   22:6    91                  NRECS := NRECS-1
1082   1   22:5    92                END;
1083   1   22:5    98              { REVERSE REF LIST }
1084   1   22:4    98              RP := ENTRY.REFLIST;
1085   1   22:4    02              ENTRY.REFLIST := NIL;
1086   1   22:4    05              WHILE RP <> NIL DO
1087   1   22:5    10                BEGIN
1088   1   22:6    10                  RQ := RP^.NEXT;
1089   1   22:6    14                  RP^.NEXT := ENTRY.REFLIST;
1090   1   22:6    18                  ENTRY.REFLIST := RP;
1091   1   22:6    21                  RP := RQ
1092   1   22:5    21                END
1093   1   22:3    24            END;
1094   1   22:2    26          IF ENTRY.LITYPE = EOFMARK THEN
1095   1   22:3    32            IF SP^.SEGKIND = HOSTSEG THEN
1096   1   22:4    38              IF (ENTRY.NEXTLC > 0)
1097   1   22:4    42              AND (ENTRY.NEXTLC <= MAXLC) THEN
1098   1   22:5    51                NEXTBASELC := ENTRY.NEXTLC
1099   1   22:4    51              ELSE
1100   1   22:5    62                ERROR('BAD HOST LC')
1101   1   22:3    76            ELSE
1102   1   22:2    80          ELSE
1103   1   22:3    82            IF ERRS = ERRCOUNT THEN
1104   1   22:4    87              BEGIN { OK...ADD TO SYMTAB }
```

```
1105   1   22:5    87              NEW(SYP);
1106   1   22:5    92              SYP^.ENTRY := ENTRY;
1107   1   22:5    99              ENTERSYM(SYP, SP^.SYMTAB)
1108   1   22:4    03            END
1109   1   22:1    05      UNTIL ENTRY.LITYPE = EOFMARK
1110   1   22:0    07    END [ READLINKINFO ] ;
1111   1   22:0    34
1112   1   22:0    34    [
1113   1   22:0    34    *  BUILDPLACES READS CODE OF SEP SEGS FROM DISK TO GENERATE
1114   1   22:0    34    *  THE PLACEREC ENTRIES FOR USE DURING PHASE3.  THE SEG IS
1115   1   22:0    34    *  READ INTO THE HEAP AND THE GROSSNESS BEGINS. ASSUME THAT
1116   1   22:0    34    *  SP <> NIL
1117   1   22:0    34    ]
1118   1   22:0    34
1119   1   26:D     1    PROCEDURE BUILDPLACES(SP: SEGP);
1120   1   26:D     2      VAR CP: CODEP; HEAP: ^INTEGER;
1121   1   26:D     4          NBYTES, NBLOCKS, NPROCS, N: INTEGER;
1122   1   26:D     8
1123   1   26:D     8        [
1124   1   26:D     8        *  PROCSRCH RECURSIVLY SEARCHES SYMTAB OF SP TO FIND
1125   1   26:D     8        *  SEPPROC AND SEPFUNC ENTRIES AND BUILD THE ACTUAL
1126   1   26:D     8        *  PLACE RECORD FOR THE LINK INFO ENTRY BY INDEXING
1127   1   26:D     8        *  THRU PROC DICT TO JTAB AND USING ENTRIC FIELD.
1128   1   26:D     8        ]
1129   1   26:D     8
1130   1   27:D     1        PROCEDURE PROCSRCH(SYMTAB: SYMP);
1131   1   27:D     2          VAR I, J: INTEGER;
1132   1   27:0     0        BEGIN
1133   1   27:1     0          IF SYMTAB <> NIL THEN
1134   1   27:2     5            BEGIN
1135   1   27:3     5              PROCSRCH(SYMTAB^.LLINK);
1136   1   27:3     9              PROCSRCH(SYMTAB^.RLINK);
1137   1   27:3    13              PROCSRCH(SYMTAB^.SLINK);
1138   1   27:3    17              WITH SYMTAB^.ENTRY DO
1139   1   27:4    22                IF LITYPE IN [SEPPROC, SEPFUNC] THEN
1140   1   27:5    31                  IF (SRCPROC <= 0) OR (SRCPROC > NPROCS) THEN
1141   1   27:6    44                    ERROR('BAD PROC #')
1142   1   27:5    57                  ELSE [ FIND BYTE PLACE IN CODE ]
1143   1   27:6    61                    BEGIN
1144   1   27:7    61                      I := NBYTES-2-2*SRCPROC;      [ POINT I AT PROC DICT ]
1145   1   27:7    73                      I := I-FETCHWORD(CP, I);      [ POINT I AT JTAB ]
```

```
1146   1   27:7    85                    IF (FETCHBYTE(CP, I) <> SRCPROC)
1147   1   27:7    96                    AND (FETCHBYTE(CP, I) <> 0) THEN
1148   1   27:8    09                      ERROR('DISAGREEING P #')
1149   1   27:7    27                    ELSE
1150   1   27:8    31                      BEGIN
1151   1   27:9    31                        J := FETCHWORD(CP, I-2)+4;
1152   1   27:9    45                        PLACE^.SRCBASE := I+2-J;
1153   1   27:9    55                        IF (PLACE^.SRCBASE < 0)
1154   1   27:9    60                        OR (J <= 0) OR (J > MAXIC) THEN
1155   1   27:0    72                          ERROR('PROC PLACE ERR')
1156   1   27:9    89                        ELSE
1157   1   27:0    93                          PLACE^.LENGTH := J
1158   1   27:8    97                      END
1159   1   27:6    04                  END
1160   1   27:2    04              END
1161   1   27:0    04          END [ PROCSRCH ] ;
1162   1   27:0    22
1163   1   26:0     0      BEGIN [ BUILDPLACES ]
1164   1   26:1     0        NBYTES := SP^.SRCFILE^.SEGTBL.DISKINFO[SP^.SRCSEG].CODELENG;
1165   1   26:1    14        NBLOCKS := (NBYTES+511) DIV 512;
1166   1   26:1    25        IF MEMAVAIL-400 < NBLOCKS*256 THEN
1167   1   26:2    39          ERROR('SEP SEG 2 BIG')
1168   1   26:1    55        ELSE
1169   1   26:2    59          BEGIN [ ALLOC SPACE IN HEAP ]
1170   1   26:3    59            MARK(HEAP);
1171   1   26:3    63            N := NBLOCKS;
1172   1   26:3    66            REPEAT
1173   1   26:4    66              NEW(CP);
1174   1   26:4    73              N := N-1
1175   1   26:3    74            UNTIL N <= 0;
1176   1   26:3    83            IF BLOCKREAD(SP^.SRCFILE^.CODE^, HEAP^, NBLOCKS,
1177   1   26:3    89                 SP^.SRCFILE^.SEGTBL.DISKINFO[SP^.SRCSEG].CODEADDR) <> NBLOCKS THEN
1178   1   26:4    11              ERROR('SEP SEG READ ERR')
1179   1   26:3    30            ELSE
1180   1   26:4    34              BEGIN
1181   1   26:5    34                CP := GETCODEP(ORD(HEAP));
1182   1   26:5    41                NPROCS := FETCHBYTE(CP, NBYTES-1);
1183   1   26:5    51                IF (NPROCS < 0) OR (NPROCS > MAXPROC) THEN
1184   1   26:6    62                  ERROR('BAD PROC DICT')
1185   1   26:5    78                ELSE
1186   1   26:6    82                  PROCSRCH(SP^.SYMTAB)
```

```
1187   1    26:4     84            END;
1188   1    26:3     86          RELEASE(HEAP)
1189   1    26:2     88        END
1190   1    26:0     90    END [ BUILDPLACES ] ;
1191   1    26:0     06
1192   1    21:0      0 BEGIN [ PHASE2 ]
1193   1    21:0      0
1194   1    21:1      0   MARK(HEAPBASE);
1195   1    21:1      4   UNITWRITE(3, HEAPBASE^, 35);
1196   1    21:1     12
1197   1    21:1     12   [ READ LINK INFO FOR HOST SEGS ]
1198   1    21:1     12
1199   1    21:1     12   DUMPSEPS := TRUE;     [ ASSUME WE DON'T NEED SEP SEGS ]
1200   1    21:1     15   FOR S := 0 TO MAXSEG DO
1201   1    21:2     32     IF SEGINFO[S] <> NIL THEN
1202   1    21:3     45       CASE SEGINFO[S]^.SEGKIND OF
1203   1    21:3     57         LINKED:     ; [ NOTHIN ]
1204   1    21:3     59         UNITSEG:    READLINKINFO(SEGINFO[S], [PUBLREF, PRIVREF, UNITREF,
1205   1    21:4     68                                               CONSTDEF,EXTPROC, EXTFUNC]);
1206   1    21:3     78         SEPRTSEG:   READLINKINFO(SEGINFO[S], [GLOBREF, GLOBDEF, CONSTDEF,
1207   1    21:4     87                                               SEPPROC, SEPFUNC]);
1208   1    21:3     97         HOSTSEG:    READLINKINFO(SEGINFO[S], [PUBLDEF, CONSTDEF,
1209   1    21:4     06                                               EXTPROC, EXTFUNC]);
1210   1    21:3     16         SEGPROC:    READLINKINFO(SEGINFO[S], [EXTPROC, EXTFUNC])
1211   1    21:3     31       END [ CASES ] ;
1212   1    21:3     59
1213   1    21:3     59   [ NOW DO SEP LIST ELEMENTS ]
1214   1    21:3     59
1215   1    21:1     59   IF DUMPSEPS THEN
1216   1    21:2     62     SEPLIST := NIL;
1217   1    21:1     65   SP := SEPLIST;
1218   1    21:1     68   WHILE SP <> NIL DO
1219   1    21:2     73     BEGIN
1220   1    21:3     73       READLINKINFO(SP, REFLITYPES+[GLOBDEF, CONSTDEF, SEPPROC, SEPFUNC]);
1221   1    21:3     85       SP := SP^.NEXT
1222   1    21:2     86     END;
1223   1    21:2     91
1224   1    21:2     91   [ BUILD PROC PLACE ENTRIES FOR SEP SEGS ]
1225   1    21:2     91
1226   1    21:1     91   IF SEGINFO[MASTERSEG]^.SEGKIND = SEPRTSEG THEN
1227   1    21:2     05     BUILDPLACES(SEGINFO[MASTERSEG]);
```

```
1228   1   21:2    16
1229   1   21:1    16    SP := SEPLIST;
1230   1   21:1    19    WHILE SP <> NIL DO
1231   1   21:2    24      BEGIN
1232   1   21:3    24        BUILDPLACES(SP);
1233   1   21:3    27        SP := SP^.NEXT
1234   1   21:2    28      END;
1235   1   21:1    33    IF ERRCOUNT > 0 THEN
1236   1   21:2    38      EXIT(LINKER)
1237   1   21:0    42  END [ PHASE2 ] ;
1238   1   21:0    60
1239   1   21:0    60  [$I LINK2 ]
1239   1   21:0    60  [$I LINK3A ]
1240   1   21:0    60
1241   1   21:0    60      (*****************************************************************)
1242   1   21:0    60      (*                                                               *)
1243   1   21:0    60      (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
1244   1   21:0    60      (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
1245   1   21:0    60      (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
1246   1   21:0    60      (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
1247   1   21:0    60      (*                                                               *)
1248   1   21:0    60      (*****************************************************************)
1249   1   21:0    60
1250   1   21:0    60  [
1251   1   21:0    60  *  PHASE3 OF THE LINKER DOES ALL THE REAL WORK OF CODE
1252   1   21:0    60  *  MASSAGING.  FOR EACH SEGMENT IN SEGINFO TO BE PLACED
1253   1   21:0    60  *  INTO THE OUTPUT CODE FILE, ALL REFERENCED PROCEDURES
1254   1   21:0    60  *  AND FUNCTIONS ARE FOUND, GLOBALS AND OTHER REFS ARE
1255   1   21:0    60  *  RESOLVED, AND FINALLY THE FINAL CODE SEGMENT IS BUILT.
1256   1   21:0    60  *  IN THE CASE OF A SEPRTSEG HOST (EG AN INTERPRETER), THEN
1257   1   21:0    60  *  ALL THE PROCS IN IT ARE PUT IN THE UNRESOLVED LIST AND
1258   1   21:0    60  *  THE HOST SEG IS MADE TO APPEAR AS JUST ANOTHER SEP SEG.
1259   1   21:0    60  *  THIS DRAGS ALONG ALL THE ORIGINAL PROCEDURES AND MAINTAINS
1260   1   21:0    60  *  THEIR ORIGINAL ORDERING FOR POSSIBLE ASECT INTEGRITY.
1261   1   21:0    60  ]
1262   1   21:0    60
1263   1   28:0     1  PROCEDURE PHASE3;
1264   1   28:0     1    TYPE
1265   1   28:0     1      WORKP = ^WORKREC;          [ ALL SEG WORK IS DRIVEN BY THESE LISTS ]
1266   1   28:0     1      WORKREC = RECORD
1267   1   28:0     1                  NEXT: WORKP;          [ LIST LINK ]
```

```
1268  1  28:D  1                    REFSYM,                  [ SYMTAB ENTRY OF UNRESOLVED NAME ]
1269  1  28:D  1                    DEFSYM: SYMP;            [   "      "   "  RESOLVING ENTRY ]
1270  1  28:D  1                    REFSEG,                  [ SEG REFLS POINT INTO, REFRANGE ONLY
1271  1  28:D  1                    DEFSEG: SEGP;            [ SEG WHERE DEFSYM WAS FOUND ]
1272  1  28:D  1                    CASE LITYPES OF          [ SAME AS LITYPE IN REFSYM^.ENTRY ]
1273  1  28:D  1                      SEPPREF,
1274  1  28:D  1                      SEPFREF,
1275  1  28:D  1                      GLOBREF:
1276  1  28:D  1                          (DEFPROC: WORKP);        [ WORK ITEM OF HOMEPROC ]
1277  1  28:D  1                      UNITREF:
1278  1  28:D  1                          (DEFSEGNUM: SEGRANGE);   [ RESOLVED SEG #, DEF = REF ]
1279  1  28:D  1                      PRIVREF:
1280  1  28:D  1                          (NEWOFFSET: LCRANGE);    [ NEWLY ASSIGNED BASE OFFSET ]
1281  1  28:D  1                      EXTPROC,
1282  1  28:D  1                      EXTFUNC,
1283  1  28:D  1                      SEPPROC,
1284  1  28:D  1                      SEPFUNC:
1285  1  28:D  1                          (NEEDSRCH: BOOLEAN;      [ REFS HAVEN'T BEEN FOUND ]
1286  1  28:D  1                           NEWPROC: 0..MAXPROC)    [ PROC #, COMP OR LINK CHOSEN
1287  1  28:D  1                    END [ WORKREC ] ;              [ 0 IMPLIES ADDED PROC ]
1288  1  28:D  1
1289  1  28:D  1  VAR S: SEGINDEX;
1290  1  28:D  2      SEGBASE: CODEP;   [ ADDRESS OF CURRENT SEG BEING CRUNCHED ]
1291  1  28:D  3      SEGLENG,          [ FINAL CODE SEG LENGTH FOR WRITEOUT ]
1292  1  28:D  3      NEXTBLK: INTEGER; [ NEXT AVAILABLE OUTPUT CODE BLOCK ]
1293  1  28:D  5      UPROCS,           [ UNRESOLVED EXTERNAL PROC/FUNC WORK LIST ]
1294  1  28:D  5      PROCS,            [ RESOLVED LIST OF ABOVE ITEMS ]
1295  1  28:D  5      ULOCAL,           [ UNRESOLVED LIST OF UPDATES FOR SEGINFO ENTRY ]
1296  1  28:D  5      LOCAL,            [ RESOLVED LIST OF FIXUPS THAT CAME ALONG WITH SEG ]
1297  1  28:D  5      UOTHER,           [ UNRESOLVED WORK LIST OF THINGS OTHER THAN PROCS ]
1298  1  28:D  5      OTHER: WORKP;     [ RESOLVED LIST OF ABOVE ]
1299  1  28:D 11      SEPHOST: BOOLEAN; [ FLAG FOR INTERPRETER HOST CASE (ONLY SEG #1) ]
1300  1  28:D 12      FNAME: STRING[39];[ OUTPUT CODE FILE NAME ]
1301  1  28:D 32      SEGTBL: I5SEGTBL; [ OUTPUT CODE'S SEG TABLE ]
1302  1  28:D 88      MAP: TEXT;        [ MAP TEXT OUTPUT FILE ]
1303  1  28:D 89
1304  1  28:D 89    [
1305  1  28:D 89     *  BUILDWORKLISTS IS CALLED FOR ALL SEGMENTS WHICH NEED TO
1306  1  28:D 89     *  BE COPIED, AND MAYBE NEED TO HAVE SEPPROCS OR OTHERS STUFF
1307  1  28:D 89     *  FIXED UP WITHIN THEM.  THE IDEA HERE IS TO GET A LIST
1308  1  28:D 89     *  OF PROCS AND OTHER ITEM NEEDING ATTENTION, WITH
```

```
1309   1   28:0    89      *  ALL THE SUBTLE IMPLICATIONS OF GLOBAL DEFS FALLING
1310   1   28:0    89      *  IN PROCS WHICH ARE NOT YET SELECTED FOR LINKING ETC.
1311   1   28:0    89      *  IN FACT, THREE LISTS ARE BUILT:
1312   1   28:0    89      *     THE PROCS LIST WITH ALL PROCS AND FUNC TO BE GRABBED
1313   1   28:0    89      *  FROM THE VARIOUS SEP SEGS.
1314   1   28:0    89      *     THE LOCAL LIST OF REFS IN THE ORIGINAL SEGMENT WHICH MUST
1315   1   28:0    89      *  ALL BE FIXED UP SUCH AS PUBLIC OR PRIVATE REFS IN A UNIT SEG.
1316   1   28:0    89      *     THE OTHER LIST WHICH HAS WORK ITEMS WHICH HAVE AT LEAST ONE
1317   1   28:0    89      *  REF OCCURING IN THE PROCS OR FUNCS IN THE PROCS LIST.
1318   1   28:0    89      ]
1319   1   28:0    89
1320   1   29:0     1     PROCEDURE BUILDWORKLISTS;
1321   1   29:0     1       VAR SP: SEGP;
1322   1   29:0     2           WP: WORKP;
1323   1   29:0     3
1324   1   29:0     3         [
1325   1   29:0     3         *  FINDPROCS GOES THROUGH SYMTAB AND BUILDS A LIST OF
1326   1   29:0     3         *  PROCEDURE AND FUNCTIONS WHICH OCCUR IN THE TREE AND
1327   1   29:0     3         *  WHOSE LITYPE IS IN THE OKSET.  THE RESULTING LIST
1328   1   29:0     3         *  IS NOT ORDERED IN ANY PARTICULAR FASHION.  IT IS
1329   1   29:0     3         *  CALLED TO BUILD INITIAL UPROC LIST.
1330   1   29:0     3         ]
1331   1   29:0     3
1332   1   30:0     3         FUNCTION FINDPROCS(OKSET: LISET; SYMTAB: SYMP): WORKP;
1333   1   30:0     5           VAR WORK: WORKP;
1334   1   30:0     6
1335   1   30:0     6             [
1336   1   30:0     6             *  PROCSRCH RECURSIVLY SEARCHES SUBTREES TO PICK OUT
1337   1   30:0     6             *  THOSE SYMBOLS WHICH ARE IN THE OKSET, GENERATES
1338   1   30:0     6             *  NEW WORK NODES, AND PUTS THEM INTO LOCAL WORK LIST.
1339   1   30:0     6             ]
1340   1   30:0     6
1341   1   31:0     1             PROCEDURE PROCSRCH(SYM: SYMP);
1342   1   31:0     2               VAR WP: WORKP;
1343   1   31:0     0             BEGIN
1344   1   31:1     0               IF SYM <> NIL THEN
1345   1   31:2     5                 BEGIN
1346   1   31:3     5                   PROCSRCH(SYM^.LLINK);
1347   1   31:3     9                   PROCSRCH(SYM^.RLINK);
1348   1   31:3    13                   PROCSRCH(SYM^.SLINK);
1349   1   31:3    17                   IF SYM^.ENTRY.LITYPE IN OKSET THEN
```

```
1350   1   31:4    26                BEGIN [ PLACE NEW NODE IN LIST ]
1351   1   31:5    26                  NEW(WP);
1352   1   31:5    31                  WP^.REFSYM := SYM;
1353   1   31:5    36                  WP^.REFSEG := NIL;
1354   1   31:5    41                  WP^.DEFSYM := NIL;
1355   1   31:5    46                  WP^.DEFSEG := NIL;
1356   1   31:5    51                  WP^.NEEDSRCH := TRUE;
1357   1   31:5    56                  IF SEPHOST THEN
1358   1   31:6    61                    WP^.NEWPROC := 0  [ SEE READSRCSEG! ]
1359   1   31:5    64                  ELSE
1360   1   31:6    73                    WP^.NEWPROC := SYM^.ENTRY.SRCPROC;
1361   1   31:5    85                  WP^.NEXT := WORK;
1362   1   31:5    90                  WORK := WP
1363   1   31:4    90                END
1364   1   31:2    94            END
1365   1   31:0    94        END [ PROCSRCH ] ;
1366   1   31:0    06
1367   1   30:0     0      BEGIN [ FINDPROCS ]
1368   1   30:1     0        WORK := NIL;
1369   1   30:1     3        PROCSRCH(SYMTAB);
1370   1   30:1     6        FINDPROCS := WORK
1371   1   30:0     6      END [ FINDPROCS ] ;
1372   1   30:0    22
1373   1   30:0    22      [
1374   1   30:0    22      *  FINDNEWPROCS IS CALLED TO PLACE NEW PROCEDURES INTO THE
1375   1   30:0    22      *  UPROCS WORK LIST THAT ARE NEEDED TO RESOLVE GLOBDEFS,
1376   1   30:0    22      *  SEPPREFS, AND SEPFREFS.  THE OTHER LIST IS TRAVERSED AND
1377   1   30:0    22      *  FOR EACH ELEMENT WHOSE DEFINING PROC HAS NOT BEEN ADDED
1378   1   30:0    22      *  INTO THE UPROCS LIST, THE DEFINING PROC IS LOCATED AND
1379   1   30:0    22      *  ADDED INTO UPROCS.
1380   1   30:0    22      ]
1381   1   30:0    22
1382   1   32:D     1      PROCEDURE FINDNEWPROCS;
1383   1   32:D     1        VAR WP, WP1: WORKP;
1384   1   32:D     3            PNUM: INTEGER;
1385   1   32:D     4
1386   1   32:D     4          [
1387   1   32:D     4          *  FINDNADD FINDS THE PROCEDURE NUMBERED PNUM IN THE
1388   1   32:D     4          *  SYMBOL TABLE SYMTAB.  AN ERROR IS GIVEN IF THE
1389   1   32:D     4          *  REQUIRED PROC CANNOT BE FOUND. IT RETURNS A WORK
1390   1   32:D     4          *  NODE FOR THE PROC ONCE IT HAS BEEN FOUND.  THIS
```

```
1391    1    32:0      4
1392    1    32:0      4
1393    1    32:0      4
1394    1    32:0      4
1395    1    32:0      4
1396    1    33:0      3
1397    1    33:0      4
1398    1    33:0      4
1399    1    33:0      4
1400    1    33:0      4
1401    1    33:0      4
1402    1    33:0      4
1403    1    33:0      4
1404    1    34:0      1
1405    1    34:0      2
1406    1    34:0      0
1407    1    34:1      0
1408    1    34:2      5
1409    1    34:3      5
1410    1    34:3      9
1411    1    34:3     13
1412    1    34:3     17
1413    1    34:4     26
1414    1    34:5     35
1415    1    34:6     35
1416    1    34:6     40
1417    1    34:7     45
1418    1    34:8     45
1419    1    34:9     51
1420    1    34:0     51
1421    1    34:0     55
1422    1    34:9     59
1423    1    34:8     59
1424    1    34:7     60
1425    1    34:6     65
1426    1    34:6     70
1427    1    34:6     75
1428    1    34:6     80
1429    1    34:6     85
1430    1    34:6     90
1431    1    34:6     95
```

```
*  NODE IS ALSO ADDED INTO THE UPROCS LIST.  ANY PROCS
*  ADDED THIS WAY ARE "INVISIBLE", DRAGGED ALONG BECAUSE
*  OF GLOBAL REFS/DEFS.
]

FUNCTION FINDNADD(SYMTAB: SYMP): WORKP;

    [
    *  PROCSRCH RECURSIVLY SEARCHES THE SYM TREE LOOKING
    *  FOR THE ACTUAL SYMBOL CONTAINING PNUM.  THIS DOES
    *  MOST OF THE WORK OF FINDNADD.
    ]

    PROCEDURE PROCSRCH(SYM: SYMP);
      VAR WP: WORKP;
    BEGIN
      IF SYM <> NIL THEN
        BEGIN
          PROCSRCH(SYM^.LLINK);
          PROCSRCH(SYM^.RLINK);
          PROCSRCH(SYM^.SLINK);
          IF SYM^.ENTRY.LITYPE IN [SEPPROC, SEPFUNC] THEN
            IF SYM^.ENTRY.SRCPROC = PNUM THEN
              BEGIN
                WP := UPROCS;
                WHILE WP <> NIL DO
                  BEGIN
                    IF WP^.REFSYM = SYM THEN
                      BEGIN
                        FINDNADD := WP;
                        EXIT(FINDNADD)
                      END;
                    WP := WP^.NEXT
                  END;
                NEW(WP);
                WP^.REFSYM := SYM;
                WP^.REFSEG := NIL;
                WP^.DEFSYM := NIL;
                WP^.DEFSEG := NIL;
                WP^.NEEDSRCH := TRUE;
                WP^.NEWPROC := 0;
```

```
1432   1   34:6    05                        WP^.NEXT := UPROCS;
1433   1   34:6    10                        UPROCS := WP;
1434   1   34:6    14                        FINDNADD := WP;
1435   1   34:6    18                        EXIT(FINDNADD)
1436   1   34:5    22                      END
1437   1   34:2    22                  END
1438   1   34:0    22              END [ PROCSRCH ] ;
1439   1   34:0    36
1440   1   33:0     0          BEGIN [ FINDNADD ]
1441   1   33:1     0            FINDNADD := NIL;
1442   1   33:1     3            PROCSRCH(SYMTAB);
1443   1   33:1     6            [ IF WE GET HERE THEN DIDNT FIND IT ]
1444   1   33:1     6            ERROR('MISSING PROC')
1445   1   33:0    21          END [ FINDNADD ] ;
1446   1   33:0    36
1447   1   32:0     0      BEGIN [ FINDNEWPROCS ]
1448   1   32:1     0        WP := OTHER;       [ ASSUME ONLY GLOBREF, SEPPREF, SEPFREF IN LIST ]
1449   1   32:1     5        WHILE WP <> NIL DO
1450   1   32:2    10          BEGIN
1451   1   32:3    10            IF WP^.DEFPROC = NIL THEN
1452   1   32:4    16              BEGIN [ FIND PROC/FUNC NEEDED ]
1453   1   32:5    16                IF WP^.REFSYM^.ENTRY.LITYPE = GLOBREF THEN
1454   1   32:6    23                  PNUM := WP^.DEFSYM^.ENTRY.HOMEPROC
1455   1   32:5    25                ELSE [ ASSUME A SEP PROC/FUNC ]
1456   1   32:6    31                  PNUM := WP^.DEFSYM^.ENTRY.SRCPROC;
1457   1   32:5    37                WP1 := PROCS;
1458   1   32:5    42                WHILE WP1 <> NIL DO
1459   1   32:6    47                  IF WP^.DEFSEG = WP1^.DEFSEG THEN
1460   1   32:7    54                    IF WP1^.DEFSYM^.ENTRY.SRCPROC = PNUM THEN
1461   1   32:8    62                      BEGIN [ ALREADY GONNA BE LINKED ]
1462   1   32:9    62                        WP^.DEFPROC := WP1;
1463   1   32:9    67                        WP1 := NIL
1464   1   32:8    67                      END
1465   1   32:7    70                    ELSE
1466   1   32:8    72                      WP1 := WP1^.NEXT
1467   1   32:6    73                  ELSE
1468   1   32:7    78                    WP1 := WP1^.NEXT;
1469   1   32:5    84                IF WP^.DEFPROC = NIL THEN [ FORCIBLY LINK IT ]
1470   1   32:6    90                  WP^.DEFPROC := FINDNADD(WP^.DEFSEG^.SYMTAB)
1471   1   32:4    96              END;
1472   1   32:3    01            WP := WP^.NEXT
```

```
1473   1    32:2    02          END [ WHILE ]
1474   1    32:0    05      END [ FINDNEWPROCS ] ;
1475   1    32:0    24
1476   1    32:0    24      [
1477   1    32:0    24      *  RESOLVE REMOVES WORK ITEMS FROM INLIST, SEARCHES SYMTABS
1478   1    32:0    24      *  FOR ITS CORRESPONDING DEFINITION SYMBOL (ERROR IF NOT FOUND),
1479   1    32:0    24      *  AND MOVES THE WORK ITEM INTO THE OUTPUT LIST.  EACH FLAVOR
1480   1    32:0    24      *  OF WORK ITEM NEEDS SOME SPECIAL HANDLING TO COLLECT EXTRA
1481   1    32:0    24      *  INFO RELATED TO SPECIFIC THINGS.  IN GENERAL, DEFSYM AND
1482   1    32:0    24      *  DEFSEG ARE FILLED IN.  THE INSERT ALGORITHM IS SPECIAL FOR
1483   1    32:0    24      *  PROCEDURE TYPES TO MAKE LIFE EASIER ON REFSRCH.
1484   1    32:0    24      ]
1485   1    32:0    24
1486   1    35:D     1      PROCEDURE RESOLVE(VAR INLIST, OUTLIST: WORKP);
1487   1    35:D     3        VAR SEG: SEGRANGE;
1488   1    35:D     4            ERR: BOOLEAN;
1489   1    35:D     5            WP: WORKP;
1490   1    35:D     6
1491   1    35:D     6          [
1492   1    35:D     6          *  SEPSRCH SEQUENTIALLY SEARCH THE SYMTABS IN THE SEPLIST
1493   1    35:D     6          *  TO RESOLVE THE REFSYM OF INLIST^.  IT BASICALLY JUST
1494   1    35:D     6          *  CALLS SYMSRCH REPETIVELY AND FIXES UP DEFSYM AND
1495   1    35:D     6          *  DEFSEG FIELDS.  IF THE NAME OF THE REFSYM COULD
1496   1    35:D     6          *  NOT BE FOUND, AN ERROR IS GIVEN.
1497   1    35:D     6          ]
1498   1    35:D     6
1499   1    36:D     1          PROCEDURE SEPSRCH(OKTYPE: LITYPES);
1500   1    36:D     2            VAR SYP: SYMP;
1501   1    36:D     3                SP: SEGP;
1502   1    36:0     0          BEGIN
1503   1    36:1     0            SP := SEPLIST;
1504   1    36:1     3            WHILE SP <> NIL DO
1505   1    36:2     8              BEGIN
1506   1    36:3     8                SYP := SYMSRCH(INLIST^.REFSYM^.ENTRY.NAME,
1507   1    36:3    15                               OKTYPE, SP^.SYMTAB);
1508   1    36:3    24                IF SYP <> NIL THEN
1509   1    36:4    29                  BEGIN
1510   1    36:5    29                    INLIST^.DEFSYM := SYP;
1511   1    36:5    37                    INLIST^.DEFSEG := SP;
1512   1    36:5    45                    SP := NIL
1513   1    36:4    45                  END
```

```
1514   1   36:3    48              ELSE
1515   1   36:4    50                SP := SP^.NEXT
1516   1   36:2    51            END
1517   1   36:0    54        END [ SEPSRCH ] ;
1518   1   36:0    70
1519   1   36:0    70        [
1520   1   36:0    70        *  PROCINSERT IS CALLED TO INSERT WORK INTO THE PROCS
1521   1   36:0    70        *  LIST USING A SPECIAL SET OF SORT KEYS SO THAT COPYIN-
1522   1   36:0    70        *  PROCS WILL RUN REASONABLY FAST AND USE THE DISK
1523   1   36:0    70        *  EFFICIENTLY.  THE PROCS LIST IS SORTED BY SEGMENT,
1524   1   36:0    70        *  SRCBASE KEYS.  THE SEG ORDERING IS DICTATED BY THE
1525   1   36:0    70        *  SEPLIST, SO USER ASECTS ETC WILL RETAIN THEIR ORIGINAL
1526   1   36:0    70        *  ORDERING.
1527   1   36:0    70        ]
1528   1   36:0    70
1529   1   37:D     1        PROCEDURE PROCINSERT(WORK: WORKP);
1530   1   37:D     2          LABEL 1;
1531   1   37:D     2          VAR CRNT, PREV: WORKP;
1532   1   37:D     4              SP: SEGP;
1533   1   37:0     0        BEGIN
1534   1   37:1     0          PREV := NIL;
1535   1   37:1     3          SP := SEPLIST;
1536   1   37:1     6          WHILE SP <> OUTLIST^.DEFSEG DO
1537   1   37:2    15            IF SP = WORK^.DEFSEG THEN
1538   1   37:3    21              GOTO 1
1539   1   37:2    23            ELSE
1540   1   37:3    25              SP := SP^.NEXT;
1541   1   37:1    31          CRNT := OUTLIST;
1542   1   37:1    37          REPEAT
1543   1   37:2    37            IF CRNT^.DEFSEG = WORK^.DEFSEG THEN
1544   1   37:3    44              REPEAT
1545   1   37:4    44                IF WORK^.DEFSYM^.ENTRY.PLACE^.SRCBASE <
1546   1   37:4    49                   CRNT^.DEFSYM^.ENTRY.PLACE^.SRCBASE THEN
1547   1   37:5    57                  GOTO 1;
1548   1   37:4    59                PREV := CRNT;
1549   1   37:4    62                CRNT := CRNT^.NEXT;
1550   1   37:4    66                IF CRNT = NIL THEN
1551   1   37:5    71                  GOTO 1
1552   1   37:3    73              UNTIL CRNT^.DEFSEG <> WORK^.DEFSEG
1553   1   37:2    76            ELSE
1554   1   37:3    82              BEGIN
```

```
1555   1    37:4    82                PREV := CRNT;
1556   1    37:4    85                CRNT := CRNT^.NEXT;
1557   1    37:4    89                IF CRNT <> NIL THEN
1558   1    37:5    94                  WHILE SP <> CRNT^.DEFSEG DO
1559   1    37:6    00                    IF SP = WORK^.DEFSEG THEN
1560   1    37:7    06                      GOTO 1
1561   1    37:6    08                    ELSE
1562   1    37:7    10                      SP := SP^.NEXT
1563   1    37:3    11              END
1564   1    37:1    16          UNTIL CRNT = NIL;
1565   1    37:1    21      1:
1566   1    37:1    21          IF PREV = NIL THEN
1567   1    37:2    26            BEGIN
1568   1    37:3    26              WORK^.NEXT := OUTLIST;
1569   1    37:3    32              OUTLIST := WORK
1570   1    37:2    35            END
1571   1    37:1    37          ELSE
1572   1    37:2    39            BEGIN
1573   1    37:3    39              WORK^.NEXT := PREV^.NEXT;
1574   1    37:3    43              PREV^.NEXT := WORK
1575   1    37:2    44            END
1576   1    37:0    46        END [ PROCINSERT ] ;
1577   1    37:0    66
1578   1    35:0     0    BEGIN [ RESOLVE ]
1579   1    35:1     0      WHILE INLIST <> NIL DO
1580   1    35:2     6        BEGIN
1581   1    35:3     6          WITH INLIST^, REFSYM^.ENTRY DO
1582   1    35:4    16            CASE LITYPE OF
1583   1    35:4    20              GLOBREF:    BEGIN
1584   1    35:6    20                            SEPSRCH(GLOBDEF);
1585   1    35:6    23                            DEFPROC := NIL
1586   1    35:5    26                          END;
1587   1    35:5    30
1588   1    35:4    30              CONSTREF:   IF HOSTSP <> NIL THEN
1589   1    35:6    35                            BEGIN
1590   1    35:7    35                              DEFSYM := SYMSRCH(NAME, CONSTDEF,
1591   1    35:7    40                                                HOSTSP^.SYMTAB);
1592   1    35:7    47                              DEFSEG := HOSTSP
1593   1    35:6    50                            END;
1594   1    35:6    54
1595   1    35:4    54              PUBLREF:    IF HOSTSP <> NIL THEN
```

```
1596   1   35:6   59                    BEGIN
1597   1   35:7   59                      DEFSYM := SYMSRCH(NAME, PUBLDEF,
1598   1   35:7   64                                        HOSTSP^.SYMTAB);
1599   1   35:7   71                      DEFSEG := HOSTSP
1600   1   35:6   74                    END;
1601   1   35:6   78
1602   1   35:4   78          PRIVREF:  BEGIN
1603   1   35:6   78                    NEWOFFSET := NEXTBASELC;
1604   1   35:6   88                    NEXTBASELC := NEXTBASELC+NWORDS;
1605   1   35:6   99                    IF HOSTSP <> NIL THEN
1606   1   35:7   04                      DEFSYM := REFSYM;
1607   1   35:6   10                    DEFSEG := HOSTSP
1608   1   35:5   13                  END;
1609   1   35:4   17          EXTPROC,
1610   1   35:4   17          SEPPROC,
1611   1   35:4   17          SEPPREF:  BEGIN
1612   1   35:6   17                    SEPSRCH(SEPPROC);
1613   1   35:6   20                    IF LITYPE = SEPPREF THEN
1614   1   35:7   26                      DEFPROC := NIL;
1615   1   35:6   31                    ERR := FALSE;
1616   1   35:6   34                    IF DEFSYM <> NIL THEN
1617   1   35:7   40                      IF LITYPE = SEPPREF THEN
1618   1   35:8   46                        ERR := DEFSYM^.ENTRY.NPARAMS <> NWORDS
1619   1   35:7   50                      ELSE
1620   1   35:8   57                        ERR := DEFSYM^.ENTRY.NPARAMS <> NPARAMS;
1621   1   35:6   66                    IF ERR THEN
1622   1   35:7   69                      BEGIN
1623   1   35:8   69                        WRITE('PROC ', NAME);
1624   1   35:8   93                        ERROR(' PARAM MISMATCH')
1625   1   35:7   11                      END
1626   1   35:5   13                  END;
1627   1   35:4   15          EXTFUNC,
1628   1   35:4   15          SEPFUNC,
1629   1   35:4   15          SEPFREF:  BEGIN
1630   1   35:6   15                    SEPSRCH(SEPFUNC);
1631   1   35:6   18                    IF LITYPE = SEPFREF THEN
1632   1   35:7   24                      DEFPROC := NIL;
1633   1   35:6   29                    ERR := FALSE;
1634   1   35:6   32                    IF DEFSYM <> NIL THEN
1635   1   35:7   38                      IF LITYPE = SEPFREF THEN
1636   1   35:8   44                        ERR := DEFSYM^.ENTRY.NPARAMS <> NWORDS
```

```
1637   1   35:7   48                          ELSE
1638   1   35:8   55                            ERR := DEFSYM^.ENTRY.NPARAMS <> NPARAMS;
1639   1   35:6   64                        IF ERR THEN
1640   1   35:7   67                          BEGIN
1641   1   35:8   67                            WRITE('FUNC ', NAME);
1642   1   35:8   91                            ERROR(' PARAM MISMATCH')
1643   1   35:7   09                          END
1644   1   35:5   11                      END;
1645   1   35:5   13
1646   1   35:4   13          UNITREF:    IF UNITSRCH(HOSTFILE, NAME, SEG) = HOSTFILE THEN
1647   1   35:6   25                        BEGIN [ WILL BE FOUND IN HOST ]
1648   1   35:7   25                          DEFSYM := REFSYM;
1649   1   35:7   31                          DEFSEGNUM := SEG
1650   1   35:6   34                        END
1651   1   35:5   39                      ELSE [ "IMPOSSIBLE" ]
1652   1   35:6   41                        ERROR('UNIT ERR')
1653   1   35:4   52        END [ CASES ] ;
1654   1   35:4   92
1655   1   35:3   92      WP := INLIST;
1656   1   35:3   96      INLIST := WP^.NEXT;
1657   1   35:3   00      IF WP^.DEFSYM = NIL THEN
1658   1   35:4   06        WITH WP^.REFSYM^.ENTRY DO
1659   1   35:5   12          BEGIN
1660   1   35:6   12            CASE LITYPE OF
1661   1   35:6   16              GLOBREF:  WRITE('GLOBAL ');
1662   1   35:6   35              PUBLREF:  WRITE('PUBLIC ');
1663   1   35:6   54              CONSTREF: WRITE('CONST ');
1664   1   35:6   72              SEPPREF,
1665   1   35:6   72              EXTPROC:  WRITE('PROC ');
1666   1   35:6   89              SEPFREF,
1667   1   35:6   89              EXTFUNC:  WRITE('FUNC ')
1668   1   35:6   04            END [ CASES ] ;
1669   1   35:6   40            WRITE(NAME);
1670   1   35:6   49            ERROR(' UNDEFINED')
1671   1   35:5   62          END
1672   1   35:3   64      ELSE
1673   1   35:4   66        IF  (WP^.DEFSYM^.ENTRY.LITYPE IN [SEPPROC, SEPFUNC])
1674   1   35:4   74        AND (OUTLIST <> NIL) THEN
1675   1   35:5   81          PROCINSERT(WP)
1676   1   35:4   82        ELSE
1677   1   35:5   86          BEGIN
```

```
1678   1   35:6    86                WP^.NEXT := OUTLIST;
1679   1   35:6    90                OUTLIST := WP
1680   1   35:5    91              END
1681   1   35:2    93          END [ WHILE ]
1682   1   35:0    93      END [ RESOLVE ] ;
1683   1   35:0    18
1684   1   35:0    18      [
1685   1   35:0    18      *  REFSRCH SLOWLY GOES THROUGH ALL REFERENCE LISTS IN SYMBOLS
1686   1   35:0    18      *  WHICH ARE IN THE OKSET TO SEE IF ANY "OCCUR" WITHIN THE
1687   1   35:0    18      *  PROCEDURES/FUNCTIONS SELECTED TO BE LINKED, THAT IS CONTAINED
1688   1   35:0    18      *  IN PROCS LIST.  IT IS ASSUMED THAT PROCS IS SORTED BY DEFSEG
1689   1   35:0    18      *  SO ONLY THE PROCS BETWEEN IPL AND LPL ARE SEARCHED.
1690   1   35:0    18      *  ANY SYMBOLS WHICH HAVE ANY REFS IN SELECTED PROCS ARE GIVEN
1691   1   35:0    18      *  WORK NODES AND ARE PLACED IN THE UOTHER LIST IN NO CERTAIN
1692   1   35:0    18      *  ORDER SO RESOLVE CAN BE CALLED RIGHT AWAY.
1693   1   35:0    18      ]
1694   1   35:0    18
1695   1   38:D     1      PROCEDURE REFSRCH(OKSET: LISET; SP: SEGP);
1696   1   38:D     3        VAR LPL, IPL: WORKP;
1697   1   38:D     5            DIFFSEG: BOOLEAN;
1698   1   38:D     6
1699   1   38:D     6          [
1700   1   38:D     6          *  CHECKREFS RECURSIVLY SEARCHES SYM TREE TO KIND NAMES
1701   1   38:D     6          *  IN THE OKSET.  WHEN ONE IS FOUND, EACH OF ITS REF POINTERS
1702   1   38:D     6          *  ARE CHECKED TO SEE IF THEY FALL IN ONE OF THE PROCS
1703   1   38:D     6          *  TO-BE-LINKED (BETWEEN IPL & LPL).  IF SO, A NEW WORK ITEM
1704   1   38:D     6          *  IS GENERATED AND IT'S PUT ON THE UOTHER LIST.
1705   1   38:D     6          ]
1706   1   38:D     6
1707   1   39:D     1          PROCEDURE CHECKREFS(SYM: SYMP);
1708   1   39:D     2            LABEL 1, 2;
1709   1   39:D     2            VAR PL, WP: WORKP;
1710   1   39:D     4                I, N, REF: INTEGER;
1711   1   39:D     7                RP: REFP;
1712   1   39:0     0          BEGIN
1713   1   39:1     0            IF SYM <> NIL THEN
1714   1   39:2     5              BEGIN
1715   1   39:3     5                CHECKREFS(SYM^.LLINK);
1716   1   39:3     9                CHECKREFS(SYM^.RLINK);
1717   1   39:3    13                CHECKREFS(SYM^.SLINK);
1718   1   39:3    17                WITH SYM^.ENTRY DO
```

```
1719   1    39:4    22
1720   1    39:5    31
1721   1    39:6    31
1722   1    39:6    35
1723   1    39:6    40
1724   1    39:7    45
1725   1    39:8    45
1726   1    39:9    50
1727   1    39:0    50
1728   1    39:0    53
1729   1    39:9    54
1730   1    39:8    58
1731   1    39:9    60
1732   1    39:8    65
1733   1    39:9    65
1734   1    39:9    77
1735   1    39:9    82
1736   1    39:0    82
1737   1    39:1    86
1738   1    39:2    92
1739   1    39:3    98
1740   1    39:2    00
1741   1    39:3    02
1742   1    39:4    11
1743   1    39:5    11
1744   1    39:5    16
1745   1    39:5    21
1746   1    39:5    28
1747   1    39:5    33
1748   1    39:5    38
1749   1    39:5    43
1750   1    39:5    47
1751   1    39:4    49
1752   1    39:0    49
1753   1    39:9    50
1754   1    39:9    60
1755   1    39:9    60
1756   1    39:8    61
1757   1    39:8    70
1758   1    39:7    71
1759   1    39:5    74
```

```
IF LITYPE IN OKSET THEN
  BEGIN
    N := NREFS;
    RP := REFLIST;
    WHILE RP <> NIL DO
      BEGIN
        IF N > 8 THEN
          BEGIN
            I := 7;
            N := N-8
          END
        ELSE
          I := N-1;
        REPEAT [ FOR EACH REF ]
          REF := RP^.REFS[I];
          PL := IPL;
          REPEAT [ SEARCH PROC LIST ]
            IF PL^.NEEDSRCH THEN
              WITH PL^.DEFSYM^.ENTRY.PLACE^ DO
                IF REF < SRCBASE THEN
                  GOTO 2 [ TERMINATE PROC SEARCH ]
                ELSE
                  IF REF < SRCBASE+LENGTH THEN
                    BEGIN [ OCCURS IN PROC ]
                      NEW(WP);
                      WP^.REFSYM := SYM;
                      WP^.REFSEG := SP;
                      WP^.DEFSYM := NIL;
                      WP^.DEFSEG := NIL;
                      WP^.NEXT := UOTHER;
                      UOTHER := WP;
                      GOTO 1
                    END;
            PL := PL^.NEXT
          UNTIL PL = LPL;
        2:
          I := I-1
        UNTIL I < 0;
        RP := RP^.NEXT
      END [ WHILE ]
END
```

```
1760   1   39:2    76              END;
1761   1   39:1    76          1:
1762   1   39:0    76            END { CHECKREFS } ;
1763   1   39:0    00
1764   1   38:0     0          BEGIN { REFSRCH }
1765   1   38:1     0            IPL := NIL;
1766   1   38:1     3            LPL := PROCS;
1767   1   38:1     8            WHILE LPL <> NIL DO
1768   1   38:2    13              IF (LPL^.DEFSEG = SP)
1769   1   38:2    17              AND LPL^.NEEDSRCH THEN
1770   1   38:3    22                BEGIN
1771   1   38:4    22                  IPL := LPL;
1772   1   38:4    25                  LPL := NIL
1773   1   38:3    25                END
1774   1   38:2    28              ELSE
1775   1   38:3    30                LPL := LPL^.NEXT;
1776   1   38:1    36            IF IPL <> NIL THEN
1777   1   38:2    41              BEGIN
1778   1   38:3    41                LPL := IPL;
1779   1   38:3    44                REPEAT
1780   1   38:4    44                  DIFFSEG := LPL^.DEFSEG <> IPL^.DEFSEG;
1781   1   38:4    51                  IF NOT DIFFSEG THEN
1782   1   38:5    55                    LPL := LPL^.NEXT
1783   1   38:3    56                UNTIL DIFFSEG OR (LPL = NIL);
1784   1   38:3    66                CHECKREFS(SP^.SYMTAB);
1785   1   38:3    70                REPEAT
1786   1   38:4    70                  IPL^.NEEDSRCH := FALSE;
1787   1   38:4    75                  IPL := IPL^.NEXT
1788   1   38:3    76                UNTIL IPL = LPL
1789   1   38:2    80              END
1790   1   38:0    84          END { REFSRCH } ;
1791   1   38:0    02
1792   1   38:0    02          {
1793   1   38:0    02          *  FINDLOCALS RECURSIVLY SEARCHES THE MAIN SEGS SYMTAB TO
1794   1   38:0    02          *  PLACE ANY UNRESOLVED THINGS LIKE PUBLIC REFS IN UNIT
1795   1   38:0    02          *  SEGS INTO THE ULOCAL LIST SO THEY CAN BE FIXED UP IN
1796   1   38:0    02          *  FIXUPREFS IN ADDITION TO THE SEP PROC THINGS.
1797   1   38:0    02          }
1798   1   38:0    02
1799   1   40:D     1          PROCEDURE FINDLOCALS(SYM: SYMP);
1800   1   40:D     2            VAR WP: WORKP;
```

```
1801   1   40:0     0        BEGIN
1802   1   40:1     0          IF SYM <> NIL THEN
1803   1   40:2     5            BEGIN
1804   1   40:3     5              FINDLOCALS(SYM^.LLINK);
1805   1   40:3     9              FINDLOCALS(SYM^.RLINK);
1806   1   40:3    13              FINDLOCALS(SYM^.SLINK);
1807   1   40:3    17              IF SYM^.ENTRY.LITYPE IN [UNITREF, PUBLREF, PRIVREF] THEN
1808   1   40:4    24                BEGIN
1809   1   40:5    24                  NEW(WP);
1810   1   40:5    29                  WP^.REFSYM := SYM;
1811   1   40:5    34                  WP^.REFSEG := NIL;
1812   1   40:5    39                  WP^.DEFSYM := NIL;
1813   1   40:5    44                  WP^.DEFSEG := NIL;
1814   1   40:5    49                  WP^.NEXT := ULOCAL;
1815   1   40:5    54                  ULOCAL := WP
1816   1   40:4    54                END
1817   1   40:2    58            END
1818   1   40:0    58        END [ FINDLOCALS ] ;
1819   1   40:0    70
1820   1   29:0     0    BEGIN [ BUILDWORKLISTS ]
1821   1   29:1     0      PROCS := NIL;
1822   1   29:1     4      LOCAL := NIL;
1823   1   29:1     8      OTHER := NIL;
1824   1   29:1    12      UPROCS := NIL;
1825   1   29:1    16      ULOCAL := NIL;
1826   1   29:1    20      UOTHER := NIL;
1827   1   29:1    24      WITH SEGINFO[S]^ DO
1828   1   29:2    37        IF SEGKIND <> LINKED THEN
1829   1   29:3    43          BEGIN
1830   1   29:4    43            SEPHOST := SEGKIND = SEPRTSEG;
1831   1   29:4    50            IF SEPHOST THEN
1832   1   29:5    55              BEGIN
1833   1   29:6    55                NEXT := SEPLIST;
1834   1   29:6    60                SEPLIST := SEGINFO[S];
1835   1   29:6    73                UPROCS := FINDPROCS([SEPPROC, SEPFUNC], SYMTAB)
1836   1   29:5    81              END
1837   1   29:4    88            ELSE
1838   1   29:5    90              UPROCS := FINDPROCS([EXTPROC, EXTFUNC], SYMTAB);
1839   1   29:4    05            WHILE UPROCS <> NIL DO
1840   1   29:5    12              BEGIN
1841   1   29:6    12                RESOLVE(UPROCS, PROCS);
```

```
1842   1   29:6    20                SP := SEPLIST;
1843   1   29:6    23                WHILE SP <> NIL DO
1844   1   29:7    28                  BEGIN
1845   1   29:8    28                    REFSRCH([GLOBREF, SEPPREF, SEPFREF], SP);
1846   1   29:8    37                    SP := SP^.NEXT
1847   1   29:7    38                  END;
1848   1   29:6    43                RESOLVE(UOTHER, OTHER);
1849   1   29:6    51                FINDNEWPROCS
1850   1   29:5    51              END;
1851   1   29:4    55            IF NOT SEPHOST THEN
1852   1   29:5    61              BEGIN
1853   1   29:6    61                FINDLOCALS(SYMTAB);
1854   1   29:6    65                RESOLVE(ULOCAL, LOCAL)
1855   1   29:5    71              END;
1856   1   29:4    73            WP := PROCS;
1857   1   29:4    78            WHILE WP <> NIL DO
1858   1   29:5    83              BEGIN
1859   1   29:6    83                WP^.NEEDSRCH := TRUE;
1860   1   29:6    88                WP := WP^.NEXT
1861   1   29:5    89              END;
1862   1   29:4    94            SP := SEPLIST;
1863   1   29:4    97            WHILE SP <> NIL DO
1864   1   29:5    02              BEGIN
1865   1   29:6    02                REFSRCH([PUBLREF, PRIVREF, CONSTREF], SP);
1866   1   29:6    09                SP := SP^.NEXT
1867   1   29:5    10              END;
1868   1   29:4    15            RESOLVE(UOTHER, OTHER)
1869   1   29:3    21          END
1870   1   29:0    23      END [ BUILDWORKLISTS ] ;
1871   1   29:0    46
1872   1   29:0    46  [$I LINK3A ]
1872   1   29:0    46  [$I LINK3B ]
1873   1   29:0    46
1874   1   29:0    46      (*****************************************************************)
1875   1   29:0    46      (*                                                               *)
1876   1   29:0    46      (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
1877   1   29:0    46      (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
1878   1   29:0    46      (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
1879   1   29:0    46      (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
1880   1   29:0    46      (*                                                               *)
1881   1   29:0    46      (*****************************************************************)
```

```
1882   1   29:0    46
1883   1   29:0    46      [
1884   1   29:0    46      *  READSRCSEG DETERMINES THE FINAL SEGMENT SIZE AFTER ADDING
1885   1   29:0    46      *  IN THE EXTERNAL PROCS/FUNCS, ALLOCATES ENOUGH AREA FOR THE
1886   1   29:0    46      *  ENTIRE OUTPUT CODE SEG, READS IN THE ORIGINAL CODE (OR USES
1887   1   29:0    46      *  IDENTITY SEGMENT FOR SEPHOST SPECIAL CASE), AND SPLITS THE
1888   1   29:0    46      *  SEGDICT OFF FROM THE CODE.  FOR ALL PROCS TO-BE-LINKED, A NEW
1889   1   29:0    46      *  DESTBASE POSITION IS ASSIGNED IN SEG AND THE NEW PROC NUM IS
1890   1   29:0    46      *  SET UP IN PDICT.  THE SEGMENT NUMBER FIELD OF THE PDICT IS
1891   1   29:0    46      *  ALSO UPDATED TO THE VALUE OF S.  ALL IS READY TO COPY IN THE
1892   1   29:0    46      *  SEP PROCS/FUNCS.  THE VALUES FOR SEGBASE AND SEGLENG ARE SET
1893   1   29:0    46      *  HERE TOO.
1894   1   29:0    46      ]
1895   1   29:0    46
1896   1   41:D     1      PROCEDURE READSRCSEG;
1897   1   41:D     1        VAR ORGLENG, ADDR,
1898   1   41:D     1            ADDLENG, ADDPROCS,
1899   1   41:D     1            NEXTSPOT: INTEGER;
1900   1   41:D     6            LAST: 0..MAXPROC;
1901   1   41:D     7            WP: WORKP;
1902   1   41:D     8            LHEAP: ^INTEGER;
1903   1   41:D     9
1904   1   41:D     9           [
1905   1   41:D     9           *  READNSPLIT ARRANGES FOR THE SOURCE SEG TO BE PLACED IN
1906   1   41:D     9           *  ROOM ALLOCATED FOR SEGBASE.  THIS MAY INVOLVE DISK READ
1907   1   41:D     9           *  OR PERHAPS ONLY CREATING AN EMPTY SEGMENT.  IN ANY CASE
1908   1   41:D     9           *  SEGBASE POINTS AT LOWEST ADDR, AND NEXTSPOT IS POINTED
1909   1   41:D     9           *  AT THE NEXT PLACE CODE CAN BE COPIED INTO.  THIS IS USED
1910   1   41:D     9           *  FOR DESTBASE ASSIGNMENT IN READSRCSEG.
1911   1   41:D     9           ]
1912   1   41:D     9
1913   1   42:D     1           PROCEDURE READNSPLIT;
1914   1   42:D     1             VAR NBLOCKS, N, PDLENG,
1915   1   42:D     1                 PDDELTA, NPROCS: INTEGER;
1916   1   42:D     6                 CP0, CP1: CODEP;
1917   1   42:0     0           BEGIN
1918   1   42:1     0             NBLOCKS := (SEGLENG+511) DIV 512;
1919   1   42:1    13             IF MEMAVAIL-400 < NBLOCKS*256 THEN
1920   1   42:2    27               BEGIN
1921   1   42:3    27                 ERROR('NO MEM ROOM');
1922   1   42:3    43                 EXIT(LINKER)
```

```
1923   1   42:2    47            END;
1924   1   42:1    47          N := NBLOCKS;
1925   1   42:1    50          REPEAT
1926   1   42:1    50            [ ALLOC HEAP SPACE ]
1927   1   42:2    50            NEW(CP1);
1928   1   42:2    57            N := N-1
1929   1   42:1    58          UNTIL N <= 0;
1930   1   42:1    67          IF SEPHOST THEN
1931   1   42:2    72            BEGIN [ SET UP IDENTITY SEG ]
1932   1   42:3    72              STOREWORD(0, SEGBASE, SEGLENG-2);
1933   1   42:3    83              NEXTSPOT := 0
1934   1   42:2    83            END
1935   1   42:1    87          ELSE
1936   1   42:2    89            BEGIN [ READ FROM DISK ]
1937   1   42:3    89              NBLOCKS := (ORGLENG+511) DIV 512;
1938   1   42:3    02              IF BLOCKREAD(SEGINFO[S]^.SRCFILE^.CODE^, SEGBASE^,
1939   1   42:3    19                        NBLOCKS, ADDR) <> NBLOCKS THEN
1940   1   42:4    33                BEGIN
1941   1   42:5    33                  ERROR('SEG READ ERR');
1942   1   42:5    50                  EXIT(LINKER)
1943   1   42:4    54                END;
1944   1   42:3    54              PDDELTA := SEGLENG-ORGLENG;
1945   1   42:3    63              NPROCS := FETCHBYTE(SEGBASE, ORGLENG-1);
1946   1   42:3    77              PDLENG := NPROCS*2+2;
1947   1   42:3    84              NEXTSPOT := ORGLENG-PDLENG;
1948   1   42:3    92              CP0 := GETCODEP(ORD(SEGBASE)+ORGLENG-PDLENG);
1949   1   42:3    07              CP1 := GETCODEP(ORD(SEGBASE)+SEGLENG-PDLENG);
1950   1   42:3    22              IF CP0 <> CP1 THEN
1951   1   42:4    27                BEGIN [ MOVE PROC DICT ]
1952   1   42:5    27                  N := PDLENG;
1953   1   42:5    30                  WHILE N > 2 DO
1954   1   42:6    35                    BEGIN
1955   1   42:7    35                      STOREWORD(PDDELTA+FETCHWORD(SEGBASE, ORGLENG-N),
1956   1   42:7    49                          SEGBASE, ORGLENG-N);
1957   1   42:7    59                      N := N-2
1958   1   42:6    60                    END;
1959   1   42:5    66                  MOVERIGHT(CP0^, CP1^, PDLENG);
1960   1   42:5    73                  FILLCHAR(CP0^, PDDELTA, 0)
1961   1   42:4    79                END
1962   1   42:2    79            END
1963   1   42:0    79        END [ READNSPLIT ] ;
```

```
1964    1    42:0     98
1965    1    41:0      0      BEGIN [ READSRCSEG ]
1966    1    41:1      0        IF SEPHOST THEN
1967    1    41:2      5          ORGLENG := 2
1968    1    41:1      5        ELSE
1969    1    41:2     10          WITH SEGINFO[SJ^, SRCFILE^.SEGTBL.DISKINFO[SRCSEG] DO
1970    1    41:3     36            BEGIN
1971    1    41:4     36              ORGLENG := CODELENG;
1972    1    41:4     40              ADDR := CODEADDR
1973    1    41:3     40            END;
1974    1    41:1     44        ADDLENG := 0;
1975    1    41:1     47        ADDPROCS := 0;
1976    1    41:1     50        WP := PROCS;
1977    1    41:1     55        WHILE WP <> NIL DO
1978    1    41:2     60          BEGIN [ ADD UP FINAL SEG SIZE ]
1979    1    41:3     60            ADDLENG := ADDLENG+WP^.DEFSYM^.ENTRY.PLACE^.LENGTH;
1980    1    41:3     69            IF WP^.NEWPROC = 0 THEN
1981    1    41:4     75              ADDPROCS := ADDPROCS+1;
1982    1    41:3     80            WP := WP^.NEXT
1983    1    41:2     81          END;
1984    1    41:1     86        MARK(LHEAP);
1985    1    41:1     90        SEGBASE := GETCODEP(ORD(LHEAP));
1986    1    41:1     98        SEGLENG := ORGLENG+ADDLENG+2*ADDPROCS;
1987    1    41:1     08        IF SEGLENG <= 0 THEN
1988    1    41:2     15          BEGIN
1989    1    41:3     15            ERROR('SIZE OFLOW');
1990    1    41:3     30            EXIT(LINKER)
1991    1    41:2     34          END;
1992    1    41:1     34        READNSPLIT;
1993    1    41:1     36        LAST := FETCHBYTE(SEGBASE, SEGLENG-1);
1994    1    41:1     55        WP := PROCS;
1995    1    41:1     60        WHILE WP <> NIL DO
1996    1    41:2     65          BEGIN [ ASSIGN PLACES IN CODE SEG ]
1997    1    41:3     65            WITH WP^.DEFSYM^.ENTRY.PLACE^ DO
1998    1    41:4     71              BEGIN
1999    1    41:5     71                DESTBASE := NEXTSPOT;
2000    1    41:5     74                NEXTSPOT := NEXTSPOT+LENGTH
2001    1    41:4     75              END;
2002    1    41:3     80            IF WP^.NEWPROC = 0 THEN
2003    1    41:4     86              BEGIN [ ASSIGN NEW PROC # ]
2004    1    41:5     86                LAST := LAST+1;
```

```
2005   1   41:5    96              IF LAST > MAXPROC THEN
2006   1   41:6    03                BEGIN
2007   1   41:7    03                  ERROR('PROC NUM OFLOW');
2008   1   41:7    22                  LAST := 1
2009   1   41:6    22                END;
2010   1   41:5    30              WP^.NEWPROC := LAST
2011   1   41:4    33            END;
2012   1   41:3    40          WP := WP^.NEXT
2013   1   41:2    41        END;
2014   1   41:1    46      STOREBYTE(LAST, SEGBASE, SEGLENG-1);
2015   1   41:1    57      STOREBYTE(S, SEGBASE, SEGLENG-2)
2016   1   41:0    68    END [ READSRCSEG ] ;
2017   1   41:0    86
2018   1   41:0    86    [
2019   1   41:0    86    *  COPYINPROCS GOES THROUGH PROCS LIST AND COPIES PROCEDURE
2020   1   41:0    86    *  BODIES FROM THE SEP SEGS INTO THE DEST CODE SEGMENT INTO
2021   1   41:0    86    *  LOCATIONS SET UP IN READSRCSEG. IF ALL GOES RIGHT, WE SHOULD
2022   1   41:0    86    *  FILL DEST SEG TO THE EXACT BYTE.  THE PROC DICT IS
2023   1   41:0    86    *  UPDATED TO SHOW PROCEDURES' POSITION.
2024   1   41:0    86    ]
2025   1   41:0    86
2026   1   43:D     1    PROCEDURE COPYINPROCS;
2027   1   43:D     1      VAR CP0, CP1, PDP,
2028   1   43:D     1          JTAB, SEPBASE: CODEP;
2029   1   43:D     6          WP: WORKP;
2030   1   43:D     7          CURSP: SEGP;
2031   1   43:D     8          LHEAP: ^INTEGER;
2032   1   43:D     9
2033   1   43:D     9        [
2034   1   43:D     9        *  READSEPSEG READS THE SEP SEG IN SP ONTO THE HEAP AS
2035   1   43:D     9        *  DONE IN PHASE 2.  WE SET UP SEPBASE AND CURSP FOR
2036   1   43:D     9        *  COPYINPROCS.
2037   1   43:D     9        ]
2038   1   43:D     9
2039   1   44:D     1        PROCEDURE READSEPSEG(SP: SEGP);
2040   1   44:D     2          VAR N, NBLOCKS: INTEGER;
2041   1   44:0     0        BEGIN
2042   1   44:1     0          RELEASE(LHEAP);
2043   1   44:1     5          N := SP^.SRCFILE^.SEGTBL.DISKINFO[SP^.SRCSEG].CODELENG;
2044   1   44:1    19          NBLOCKS := (N+511) DIV 512;
2045   1   44:1    30          IF MEMAVAIL-400 < NBLOCKS*256 THEN
```

```
2046   1   44:2    44            BEGIN
2047   1   44:3    44              ERROR('OUT OF MEM');
2048   1   44:3    59              EXIT(LINKER)
2049   1   44:2    63            END;
2050   1   44:1    63          N := NBLOCKS;
2051   1   44:1    66          REPEAT
2052   1   44:2    66            NEW(SEPBASE);
2053   1   44:2    74            N := N-1
2054   1   44:1    75          UNTIL N <= 0;
2055   1   44:1    84          SEPBASE := GETCODEP(ORD(LHEAP));
2056   1   44:1    94          IF BLOCKREAD(SP^.SRCFILE^.CODE^, SEPBASE^, NBLOCKS,
2057   1   44:1    02              SP^.SRCFILE^.SEGTBL.DISKINFO[SP^.SRCSEG].CODEADDR) <> NBLOCKS THE
2058   1   44:2    24            BEGIN
2059   1   44:3    24              ERROR('SEP SEG READ ERR');
2060   1   44:3    45              EXIT(LINKER)
2061   1   44:2    49            END;
2062   1   44:1    49          CURSP := SP
2063   1   44:0    49        END [ READSEPSEG ] ;
2064   1   44:0    68
2065   1   43:0     0      BEGIN [ COPYINPROCS ]
2066   1   43:1     0        SEPBASE := NIL;
2067   1   43:1     3        CURSP := NIL;
2068   1   43:1     6        MARK(LHEAP);
2069   1   43:1    10        WP := PROCS;
2070   1   43:1    15        WHILE WP <> NIL DO
2071   1   43:2    20          WITH WP^, DEFSYM^.ENTRY DO
2072   1   43:3    29            BEGIN [ COPY IN EACH PROC ]
2073   1   43:4    29              IF CURSP <> DEFSEG THEN
2074   1   43:5    35                READSEPSEG(DEFSEG);
2075   1   43:4    39              IF TALKATIVE THEN
2076   1   43:5    42                BEGIN
2077   1   43:6    42                  WRITE('   COPYING ');
2078   1   43:6    63                  IF LITYPE = SEPPROC THEN
2079   1   43:7    69                    WRITE('PROC ')
2080   1   43:6    84                  ELSE
2081   1   43:7    86                    WRITE('FUNC ');
2082   1   43:6    01                  WRITELN(NAME)
2083   1   43:5    16                END;
2084   1   43:4    16              CP0 := GETCODEP(ORD(SEPBASE)+PLACE^.SRCBASE);
2085   1   43:4    27              CP1 := GETCODEP(ORD(SEGBASE)+PLACE^.DESTBASE);
2086   1   43:4    40              MOVELEFT(CP0^, CP1^, PLACE^.LENGTH);
```

```
2087   1   43:4    49            JTAB := GETCODEP(ORD(SEGBASE)+PLACE^.DESTBASE+PLACE^.LENGTH-2);
2088   1   43:4    68            IF FETCHBYTE(JTAB, 0) <> 0 THEN
2089   1   43:5    78              STOREBYTE(NEWPROC, JTAB, 0);
2090   1   43:4    84            PDP := GETCODEP(ORD(SEGBASE)+SEGLENG-2*NEWPROC-2);
2091   1   43:4    04            STOREWORD(ORD(PDP)-ORD(JTAB), PDP, 0);
2092   1   43:4    11            WP := NEXT
2093   1   43:3    11          END;
2094   1   43:1    17      RELEASE(LHEAP)
2095   1   43:0    19    END [ COPYINPROCS ] ;
2096   1   43:0    38
2097   1   43:0    38    [
2098   1   43:0    38    *  FIXUPREFS IS CALLED TO SEARCH THROUGH REFLISTS AND FIX
2099   1   43:0    38    *  OPERAND FIELDS OF P-CODE AND NATIVE CODE TO REFER TO THE
2100   1   43:0    38    *  RESOLVED VALUES.  IF FIXALLREFS IS TRUE, THEN ALL POINTERS
2101   1   43:0    38    *  IN THE REF LISTS ARE USED, OTHERWISE THE REFERENCE POINTERS
2102   1   43:0    38    *  ARE CHECKED TO SEE IF THEY OCCUR IN THE PROCS TO-BE-LINKED.
2103   1   43:0    38    ]
2104   1   43:0    38
2105   1   45:D     1    PROCEDURE FIXUPREFS(WORK: WORKP; FIXALLREFS: BOOLEAN);
2106   1   45:D     3      VAR N, I, REF, VAL: INTEGER;
2107   1   45:D     7          WP, WP1: WORKP;
2108   1   45:D     9          RP: REFP;
2109   1   45:D    10          SKIPIT: BOOLEAN;
2110   1   45:0     0    BEGIN
2111   1   45:1     0      WHILE WORK <> NIL DO
2112   1   45:2     5        WITH WORK^, REFSYM^.ENTRY DO
2113   1   45:3    14          BEGIN [ FOR EACH WORK ITEM ]
2114   1   45:3    14                [ FIGURE RESOLVE VAL ]
2115   1   45:4    14            CASE LITYPE OF
2116   1   45:4    18              SEPPREF,
2117   1   45:4    18              SEPFREF:  VAL := DEFPROC^.NEWPROC;
2118   1   45:4    25              UNITREF:  VAL := DEFSEGNUM;
2119   1   45:4    31              CONSTREF: VAL := DEFSYM^.ENTRY.CONSTVAL;
2120   1   45:4    39              GLOBREF:  VAL := DEFSYM^.ENTRY.ICOFFSET+
2121   1   45:5    43                               DEFPROC^.DEFSYM^.ENTRY.PLACE^.DESTBASE;
2122   1   45:4    54              PUBLREF,
2123   1   45:4    54              PRIVREF:  BEGIN
2124   1   45:6    54                          IF LITYPE = PRIVREF THEN
2125   1   45:7    60                            VAL := NEWOFFSET
2126   1   45:6    60                          ELSE
2127   1   45:7    66                            VAL := DEFSYM^.ENTRY.BASEOFFSET;
```

```
2128   1    45:6    72                    IF FORMAT = WORD THEN
2129   1    45:7    78                      VAL := (VAL-1)*2+MSDELTA
2130   1    45:6    83                    ELSE [ ASSUME BIG ]
2131   1    45:7    89                      IF VAL < 0 THEN
2132   1    45:8    94                        ERROR('ADDR OFLOW')
2133   1    45:5    07                  END
2134   1    45:4    09          END;
2135   1    45:4    46          N := NREFS;
2136   1    45:4    50          RP := REFLIST;
2137   1    45:4    55          WHILE RP <> NIL DO
2138   1    45:5    60            BEGIN
2139   1    45:6    60              IF N > 8 THEN
2140   1    45:7    65                BEGIN
2141   1    45:8    65                  I := 7;
2142   1    45:8    68                  N := N-8
2143   1    45:7    69                END
2144   1    45:6    73              ELSE
2145   1    45:7    75                I := N-1;
2146   1    45:6    80              REPEAT
2147   1    45:7    80                REF := RP^.REFS[I];
2148   1    45:7    92                SKIPIT := NOT FIXALLREFS;
2149   1    45:7    96                IF SKIPIT THEN
2150   1    45:8    99                  BEGIN [ SEE IF PERTINENT ]
2151   1    45:9    99                    WP := NIL;
2152   1    45:9    02                    WP1 := PROCS;
2153   1    45:9    07                    WHILE WP1 <> NIL DO
2154   1    45:0    12                      IF WP1^.DEFSEG = REFSEG THEN
2155   1    45:1    19                        BEGIN [ FIND MATCHING SEG ]
2156   1    45:2    19                          WP := WP1;
2157   1    45:2    22                          WP1 := NIL
2158   1    45:1    22                        END
2159   1    45:0    25                      ELSE
2160   1    45:1    27                        WP1 := WP1^.NEXT;
2161   1    45:9    33                    WHILE (WP <> NIL) AND SKIPIT DO
2162   1    45:0    40                      IF WP^.DEFSEG = REFSEG THEN
2163   1    45:1    47                        WITH WP^.DEFSYM^.ENTRY.PLACE^ DO
2164   1    45:2    53                          IF REF >= SRCBASE THEN
2165   1    45:3    59                            IF REF < SRCBASE+LENGTH THEN
2166   1    45:4    68                              BEGIN
2167   1    45:5    68                                REF := REF-SRCBASE+DESTBASE;
2168   1    45:5    77                                SKIPIT := FALSE
```

```
2169   1   45:4    77                                  END
2170   1   45:3    80                                ELSE
2171   1   45:4    82                                  WP := WP^.NEXT
2172   1   45:2    83                              ELSE
2173   1   45:3    88                                WP := NIL
2174   1   45:0    88                            ELSE
2175   1   45:1    93                              WP := NIL
2176   1   45:8    93                        END;
2177   1   45:7    98                      IF NOT SKIPIT THEN
2178   1   45:8    02                        CASE FORMAT OF [ FIX UP THIS REF ]
2179   1   45:8    06                          WORD:  STOREWORD(VAL+FETCHWORD(SEGBASE, REF),
2180   1   45:9    16                                                  SEGBASE, REF);
2181   1   45:8    24                          BYTE:  STOREBYTE(VAL, SEGBASE, REF);
2182   1   45:8    33                          BIG:   STOREBIG(VAL, SEGBASE, REF)
2183   1   45:8    38                        END;
2184   1   45:7    56                      I := I-1
2185   1   45:6    57                    UNTIL I < 0;
2186   1   45:6    66                    RP := RP^.NEXT
2187   1   45:5    67                  END;
2188   1   45:4    72                WORK := NEXT
2189   1   45:3    72              END
2190   1   45:0    76      END [ FIXUPREFS ] ;
2191   1   45:0    04
2192   1   45:0    04      [
2193   1   45:0    04      *  WRITETOCODE TAKES THE FINALIZED DESTSEG AND PUTS IT IN
2194   1   45:0    04      *  THE OUTPUT CODE FILE.  THIS ALSO INVOLVES SETTING UP VALUES
2195   1   45:0    04      *  IN THE FINAL SEGTABLE FOR WRITEOUT JUST BEFORE LOCKING IT.
2196   1   45:0    04      ]
2197   1   45:0    04
2198   1   46:D     1      PROCEDURE WRITETOCODE;
2199   1   46:D     1        VAR NBLOCKS: INTEGER;
2200   1   46:D     2            JTAB: CODEP;
2201   1   46:0     0      BEGIN
2202   1   46:1     0        IF HOSTSP = SEGINFO[S] THEN
2203   1   46:2    15          BEGIN [ FIX UP BASELC ]
2204   1   46:3    15            JTAB := GETCODEP(ORD(SEGBASE)+SEGLENG-4);
2205   1   46:3    30            JTAB := GETCODEP(ORD(JTAB)-FETCHWORD(JTAB, 0));
2206   1   46:3    44            STOREWORD(NEXTBASELC*2-6, JTAB, -8)
2207   1   46:2    52          END;
2208   1   46:1    54        WITH SEGINFO[S]^, SEGTBL DO
2209   1   46:2    67          BEGIN
```

```
2210   1   46:3    67          NBLOCKS := (SEGLENG+511) DIV 512;
2211   1   46:3    80          IF BLOCKWRITE(CODE, SEGBASE^, NBLOCKS, NEXTBLK) <> NBLOCKS THEN
2212   1   46:4    00            BEGIN
2213   1   46:5    00              ERROR('CODE WRITE ERR');
2214   1   46:5    19              EXIT(LINKER)
2215   1   46:4    23            END;
2216   1   46:3    23          DISKINFO[S].CODEADDR := NEXTBLK;
2217   1   46:3    38          DISKINFO[S].CODELENG := SEGLENG;
2218   1   46:3    55          SEGNAME[S] := SRCFILE^.SEGTBL.SEGNAME[SRCSEG];
2219   1   46:3    79          SEGKIND[S] := LINKED;
2220   1   46:3    93          NEXTBLK := NEXTBLK+NBLOCKS
2221   1   46:2    96        END
2222   1   46:0    01    END [ WRITETOCODE ] ;
2223   1   46:0    14
2224   1   46:0    14    [
2225   1   46:0    14    *  LINKSEGMENT IS CALLED FOR EACH SEGMENT TO BE PLACED INTO
2226   1   46:0    14    *  THE FINAL CODE FILE.  THE GLOBAL VAR S HAS THE SEGINFO INDEX
2227   1   46:0    14    *  PERTAINING TO THE SEGMENT, AND ALL THE OTHER PROCEDURES OF
2228   1   46:0    14    *  PHASE 3 ARE CALLED FROM HERE.  THIS PROC FACILITATES LINKING
2229   1   46:0    14    *  THE MASTER SEG SEPARATLY FROM THE OTHER SEGS TO ENSURE THAT
2230   1   46:0    14    *  THE DATASZ OF THE OUTER BLOCK CORRECTLY REFLECTS THE NUMBER
2231   1   46:0    14    *  OF PRIVREF WORDS ALLOCATED BY RESOLVE.
2232   1   46:0    14    ]
2233   1   46:0    14
2234   1   47:D     1    PROCEDURE LINKSEGMENT;
2235   1   47:D     1
2236   1   47:D     1        [
2237   1   47:D     1        *  WRITEMAP IS CALLED FOR EACH SEG TO WRITE SOME
2238   1   47:D     1        *  INFO INTO MAP FILE.
2239   1   47:D     1        ]
2240   1   47:D     1
2241   1   48:D     1        PROCEDURE WRITEMAP;
2242   1   48:D     1          VAR WP: WORKP;
2243   1   48:D     2              B: BOOLEAN;
2244   1   48:0     0        BEGIN
2245   1   48:1     0          WITH SEGINFO[S]^ DO
2246   1   48:2    13            WRITELN(MAP, 'SEG # ',S,', ', SRCFILE^.SEGTBL.SEGNAME[SRCSEG]);
2247   1   48:1    81          WP := PROCS;
2248   1   48:1    86          IF WP <> NIL THEN
2249   1   48:2    91            WRITELN(MAP, '   SEP PROCS');
2250   1   48:1    21          WHILE WP <> NIL DO
```

```
2251  1  48:2  26          WITH WP^.DEFSYM^.ENTRY DO
2252  1  48:3  32            BEGIN
2253  1  48:4  32              WRITE(MAP, '      ', NAME);
2254  1  48:4  59              IF LITYPE = SEPPROC THEN
2255  1  48:5  65                WRITE(MAP, ' PROC')
2256  1  48:4  81              ELSE
2257  1  48:5  83                WRITE(MAP, ' FUNC');
2258  1  48:4  99              WRITE(MAP, ' # ', WP^.NEWPROC: 3);
2259  1  48:4  23              WRITE(MAP, '    BASE =', PLACE^.DESTBASE: 6);
2260  1  48:4  55              WRITE(MAP, '    LENG =', PLACE^.LENGTH: 5);
2261  1  48:4  87              WRITELN(MAP);
2262  1  48:4  94              WP := WP^.NEXT
2263  1  48:3  95            END;
2264  1  48:1  00        FOR B := FALSE TO TRUE DO
2265  1  48:2  11          BEGIN
2266  1  48:3  11            IF B THEN
2267  1  48:4  14              BEGIN
2268  1  48:5  14                WP := OTHER;
2269  1  48:5  19                IF WP <> NIL THEN
2270  1  48:6  24                  WRITELN(MAP, '   SEP PROC REFS')
2271  1  48:4  58              END
2272  1  48:3  58            ELSE
2273  1  48:4  60              BEGIN
2274  1  48:5  60                WP := LOCAL;
2275  1  48:5  65                IF WP <> NIL THEN
2276  1  48:6  70                  WRITELN(MAP, '   LOCAL SEG REFS')
2277  1  48:4  05              END;
2278  1  48:3  05            WHILE WP <> NIL DO
2279  1  48:4  10              WITH WP^.DEFSYM^.ENTRY DO
2280  1  48:5  16                BEGIN
2281  1  48:6  16                  WRITE(MAP, '      ', NAME);
2282  1  48:6  43                  CASE LITYPE OF
2283  1  48:6  47                    SEPPROC,
2284  1  48:6  47                    SEPFUNC:    ;
2285  1  48:6  49                    PUBLDEF:    WRITE(MAP, ' PUBLIC LC =', BASEOFFSET: 5);
2286  1  48:6  84                    CONSTDEF:   WRITE(MAP, ' CONST VAL =', CONSTVAL: 6);
2287  1  48:6  19                    PRIVREF:    WRITE(MAP, ' PRIVAT LC =', WP^.NEWOFFSET: 5);
2288  1  48:6  54                    UNITREF:    WRITE(MAP, ' UNIT SEG# =', WP^.DEFSEGNUM: 3);
2289  1  48:6  89                    GLOBDEF:    WRITE(MAP, ' GLOB DEF IN ',
2290  1  48:7  13                                   WP^.DEFPROC^.DEFSYM^.ENTRY.NAME,
2291  1  48:7  27                                   ' @', ICOFFSET: 5)
```

```
2292   1   48:6    50                      END;
2293   1   48:6    84                      WRITELN(MAP);
2294   1   48:6    91                      WP := WP^.NEXT
2295   1   48:5    92                    END
2296   1   48:2    95              END;
2297   1   48:1    04            WRITELN(MAP)
2298   1   48:0    11          END [ WRITEMAP ] ;
2299   1   48:0    40
2300   1   47:0     0      BEGIN [ LINKSEGMENT ]
2301   1   47:1     0        SEPHOST := FALSE;
2302   1   47:1     4        SEGBASE := NIL;
2303   1   47:1     8        SEGLENG := 0;
2304   1   47:1    12        IF TALKATIVE THEN
2305   1   47:2    15          WITH SEGINFO[S]^ DO
2306   1   47:3    28            WRITELN('LINKING ',
2307   1   47:3    46                    SRCFILE^.SEGTBL.SEGNAME[SRCSEG], ' # ', S);
2308   1   47:1    94        BUILDWORKLISTS;
2309   1   47:1    96        IF ERRCOUNT = 0 THEN
2310   1   47:2    01          BEGIN
2311   1   47:3    01            READSRCSEG;
2312   1   47:3    03            IF MAPNAME <> '' THEN
2313   1   47:4    12              WRITEMAP;
2314   1   47:3    14            COPYINPROCS;
2315   1   47:3    16            FIXUPREFS(LOCAL, TRUE);
2316   1   47:3    22            FIXUPREFS(OTHER, FALSE);
2317   1   47:3    28            WRITETOCODE
2318   1   47:2    28          END;
2319   1   47:1    30        IF SEPHOST THEN
2320   1   47:2    35          SEPLIST := SEGINFO[S]^.NEXT;
2321   1   47:1    49        RELEASE(HEAPBASE)
2322   1   47:0    51      END [ LINKSEGMENT ] ;
2323   1   47:0    66
2324   1   28:0     0  BEGIN [ PHASE3 ]
2325   1   28:1     0    IF NOT USEWORKFILE THEN
2326   1   28:2    17      BEGIN
2327   1   28:3    17        WRITE('OUTPUT FILE? ');
2328   1   28:3    40        READLN(FNAME);
2329   1   28:3    55        USEWORKFILE := FNAME = ''
2330   1   28:2    57      END;
2331   1   28:1    64    IF USEWORKFILE THEN
2332   1   28:2    67      REWRITE(CODE, '*SYSTEM.WRK.CODE[*]')
```

```
2333   1   28:1    96   ELSE
2334   1   28:2    98     REWRITE(CODE, FNAME);
2335   1   28:1    07   IF IORESULT <> 0 THEN
2336   1   28:2    13     BEGIN
2337   1   28:3    13       ERROR('CODE OPEN ERR');
2338   1   28:3    31       EXIT(LINKER)
2339   1   28:2    35     END;
2340   1   28:1    35   NEXTBLK := 1;
2341   1   28:1    38   [ CLEAR OUTPUT SEG TABLE ]
2342   1   28:1    38   FILLCHAR(SEGTBL, SIZEOF(SEGTBL), 0);
2343   1   28:1    47   WITH SEGTBL DO
2344   1   28:2    47     FOR S := 0 TO MAXSEG DO
2345   1   28:3    67       BEGIN
2346   1   28:4    67         SEGNAME[S] := '        ';
2347   1   28:4    88         SEGKIND[S] := LINKED
2348   1   28:3    97       END;
2349   1   28:1    06   IF MAPNAME <> '' THEN
2350   1   28:2    15     BEGIN
2351   1   28:3    15       REWRITE(MAP, MAPNAME);
2352   1   28:3    25       IF IORESULT <> 0 THEN
2353   1   28:4    31         BEGIN
2354   1   28:5    31           WRITELN('CAN''T OPEN ', MAPNAME);
2355   1   28:5    67           MAPNAME := ''
2356   1   28:4    69         END
2357   1   28:3    74       ELSE
2358   1   28:4    76         BEGIN
2359   1   28:5    76           WRITE(MAP, 'LINK MAP FOR ');
2360   1   28:5    99           IF HOSTSP <> NIL THEN
2361   1   28:6    04             WRITELN(MAP, HOSTSP^.SRCFILE^.SEGTBL.SEGNAME[HOSTSP^.SRCSEG])
2362   1   28:5    29           ELSE
2363   1   28:6    31             WRITELN(MAP, 'ASSEM HOST');
2364   1   28:5    57           WRITELN(MAP)
2365   1   28:4    63         END
2366   1   28:2    63     END;
2367   1   28:1    63   MARK(HEAPBASE);
2368   1   28:1    67   UNITWRITE(3, HEAPBASE^, 35);
2369   1   28:1    75   [ LINK ALL BUT HOST ]
2370   1   28:1    75   FOR S := 0 TO MAXSEG DO
2371   1   28:2    95     IF (SEGINFO[S] <> NIL)
2372   1   28:2    06     AND (SEGINFO[S] <> HOSTSP) THEN
2373   1   28:3    20       LINKSEGMENT;
```

```
2374   1   28:3    29   [ LINK HOST LAST! ]
2375   1   28:1    29   IF HOSTSP <> NIL THEN
2376   1   28:2    34     BEGIN
2377   1   28:3    34       S := MASTERSEG;
2378   1   28:3    40       LINKSEGMENT
2379   1   28:2    40     END;
2380   1   28:1    42   IF FLIPPED THEN FLIPTABLE(SEGTBL); [ RESTORE BYTE-FLIPPED STATE ]
2381   1   28:1    51   IF BLOCKWRITE(CODE, SEGTBL, 1, 0) <> 1 THEN
2382   1   28:2    68     ERROR('CODE WRITE ERR');
2383   1   28:1    87   IF ERRCOUNT = 0 THEN
2384   1   28:2    92     BEGIN [ FINAL CLEANUP ]
2385   1   28:3    92       CLOSE(CODE, LOCK);
2386   1   28:3    98       IF USEWORKFILE THEN
2387   1   28:4    01         WITH USERINFO DO
2388   1   28:5    01           BEGIN
2389   1   28:6    01             GOTCODE := TRUE;
2390   1   28:6    05             CODEVID := SYVID;
2391   1   28:6    13             CODETID := 'SYSTEM.WRK.CODE'
2392   1   28:5    16           END;
2393   1   28:3    36       IF MAPNAME <> '' THEN
2394   1   28:4    45         BEGIN
2395   1   28:5    45           IF HOSTSP <> NIL THEN
2396   1   28:6    50             WRITELN(MAP, 'NEXT BASE LC = ', NEXTBASELC);
2397   1   28:5    89           CLOSE(MAP, LOCK)
2398   1   28:4    96         END
2399   1   28:2    96     END
2400   1   28:2    96
2401   1   28:0    96 END [ PHASE3 ] ;
2402   1   28:0    22
2403   1   28:0    22 [$I LINK3B ]
2404   1   28:0    22
2405   1    1:0     0 BEGIN [ LINKER ]
2406   1    1:1     0   PHASE1;
2407   1    1:1    17   PHASE2;
2408   1    1:1    19   PHASE3;
2409   1    1:1    21   UNITCLEAR(3)
2410   1    1:0    22 END [ LINKER ] ;
2411   1    1:0    98
2412   0    1:0     0 BEGIN END.
```

AS OF THE PRINTING OF THIS BOOK,
THE ASSEMBLER WAS NOT LISTED.

THE ASSEMBLER MAY BE AVAILABLE
AT A LATER DATE IN A SUPPLIMENT
AT ADDITIONAL COST.

THANK YOU FOR YOUR PATIENCE. ED.

```
 1    1     1:0     1 (*$L PRINTER: *)
 2    1     1:0     1 (*$U-*)
 3    1     1:0     1
 4    1     1:0     1 (*$I GLOBALS.TEXT*)
 5    1     1:0     1 (*$U-*)
 6    1     1:0     1     [$S+]
 7    1     1:0     1     (*****************************************************************)
 8    1     1:0     1     (*                                                               *)
 9    1     1:0     1     (*  COPYRIGHT (C) 1978 REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
10    1     1:0     1     (*  PERMISSION TO COPY OR DISTRIBUTE THIS SOFTWARE OR DOCUMEN-   *)
11    1     1:0     1     (*  TATION IN HARD OR SOFT COPY GRANTED ONLY BY WRITTEN LICENSE  *)
12    1     1:0     1     (*  OBTAINED FROM THE INSTITUTE FOR INFORMATION SYSTEMS.         *)
13    1     1:0     1     (*                                                               *)
14    1     1:0     1     (*****************************************************************)
15    0     1:0     1 PROGRAM PASCALSYSTEM;
16    0     1:0     1
17    0     1:0     1 (*************************************************)
18    0     1:0     1 (*                                               *)
19    0     1:0     1 (*    UCSD PASCAL OPERATING SYSTEM               *)
20    0     1:0     1 (*                                               *)
21    0     1:0     1 (*    RELEASE LEVEL:  I.3   AUGUST, 1977         *)
22    0     1:0     1 (*                    I.4   JANUARY, 1978        *)
23    0     1:0     1 (*                    I.5   SEPTEMBER, 1978      *)
24    0     1:0     1 (*                    II.0  FEBRUARY, 1978 BD    *)
25    0     1:0     1 (*                                               *)
26    0     1:0     1 (*    WRITTEN BY ROGER T. SUMNER                 *)
27    0     1:0     1 (*    WINTER 1977                                *)
28    0     1:0     1 (*                                               *)
29    0     1:0     1 (*    INSTITUTE FOR INFORMATION SYSTEMS          *)
30    0     1:0     1 (*    UC SAN DIEGO, LA JOLLA, CA                 *)
31    0     1:0     1 (*                                               *)
32    0     1:0     1 (*    KENNETH L. BOWLES, DIRECTOR                *)
33    0     1:0     1 (*                                               *)
34    0     1:0     1 (*************************************************)
35    0     1:0     1
36    0     1:0     1 CONST
37    0     1:0     1       MMAXINT = 32767;    (*MAXIMUM INTEGER VALUE*)
38    0     1:0     1       MAXUNIT = 12;       (*MAXIMUM PHYSICAL UNIT # FOR UREAD*)
39    0     1:0     1       MAXDIR = 77;        (*MAX NUMBER OF ENTRIES IN A DIRECTORY*)
40    0     1:0     1       VIDLENG = 7;        (*NUMBER OF CHARS IN A VOLUME ID*)
```

```
41   0    1:0     1      TIDLENG = 15;        (*NUMBER OF CHARS IN TITLE ID*)
42   0    1:0     1      MAXSEG = 15;         (*MAX CODE SEGMENT NUMBER*)
43   0    1:0     1      FBLKSIZE = 512;      (*STANDARD DISK BLOCK LENGTH*)
44   0    1:0     1      DIRBLK = 2;          (*DISK ADDR OF DIRECTORY*)
45   0    1:0     1      AGELIMIT = 300;      (*MAX AGE FOR GDIRP...IN TICKS*)
46   0    1:0     1      EOL = 13;            (*END-OF-LINE...ASCII CR*)
47   0    1:0     1      DLE = 16;            (*BLANK COMPRESSION CODE*)
48   0    1:0     1      NAME_LEN = 23;       [LENGTH OF CONCAT(VIDLENG,':',TIDLENG)]
49   0    1:0     1      FILL_LEN = 11;       [MAXIMUM # OF NULLS IN FILLER]
50   0    1:0     1
51   0    1:0     1  TYPE
52   0    1:0     1
53   0    1:0     1      IORSLTWD = (INOERROR,IBADBLOCK,IBADUNIT,IBADMODE,ITIMEOUT,
54   0    1:0     1                  ILOSTUNIT,ILOSTFILE,IBADTITLE,INOROOM,INOUNIT,
55   0    1:0     1                  INOFILE,IDUPFILE,INOTCLOSED,INOTOPEN,IBADFORMAT,
56   0    1:0     1                  ISTRGOVFL);
57   0    1:0     1
58   0    1:0     1                                        (*COMMAND STATES...SEE GETCMD*)
59   0    1:0     1
60   0    1:0     1      CMDSTATE = (HALTINIT,DEBUGCALL,
61   0    1:0     1                  UPROGNOU,UPROGUOK,SYSPROG,
62   0    1:0     1                  COMPONLY,COMPANDGO,COMPDEBUG,
63   0    1:0     1                  LINKANDGO,LINKDEBUG);
64   0    1:0     1
65   0    1:0     1                                        (*CODE FILES USED IN GETCMD*)
66   0    1:0     1
67   0    1:0     1      SYSFILE = (ASSMBLER,COMPILER,EDITOR,FILER,LINKER);
68   0    1:0     1
69   0    1:0     1                                        (*ARCHIVAL INFO...THE DATE*)
70   0    1:0     1
71   0    1:0     1      DATEREC = PACKED RECORD
72   0    1:0     1                  MONTH: 0..12;        (*0 IMPLIES DATE NOT MEANINGFUL*)
73   0    1:0     1                  DAY: 0..31;          (*DAY OF MONTH*)
74   0    1:0     1                  YEAR: 0..100         (*100 IS TEMP DISK FLAG*)
75   0    1:0     1                END (*DATEREC*) ;
76   0    1:0     1
77   0    1:0     1                                        (*VOLUME TABLES*)
78   0    1:0     1      UNITNUM = 0..MAXUNIT;
79   0    1:0     1      VID = STRING[VIDLENG];
80   0    1:0     1
81   0    1:0     1                                        (*DISK DIRECTORIES*)
```

```
 82   0    1:0     1      DIRRANGE = 0..MAXDIR;
 83   0    1:0     1      TID = STRING[TIDLENG];
 84   0    1:0     1      FULL_ID = STRING[NAME_LEN];
 85   0    1:0     1
 86   0    1:0     1      FILE_TABLE = ARRAY [SYSFILE] OF FULL_ID;
 87   0    1:0     1
 88   0    1:0     1      FILEKIND = (UNTYPEDFILE,XDSKFILE,CODEFILE,TEXTFILE,
 89   0    1:0     1                  INFOFILE,DATAFILE,GRAFFILE,FOTOFILE,SECUREDIR);
 90   0    1:0     1
 91   0    1:0     1      DIRENTRY = PACKED RECORD
 92   0    1:0     1                   DFIRSTBLK: INTEGER;    (*FIRST PHYSICAL DISK ADDR*)
 93   0    1:0     1                   DLASTBLK: INTEGER;     (*POINTS AT BLOCK FOLLOWING*)
 94   0    1:0     1                   CASE DFKIND: FILEKIND OF
 95   0    1:0     1                     SECUREDIR,
 96   0    1:0     1                     UNTYPEDFILE: (*ONLY IN DIR[0]...VOLUME INFO*)
 97   0    1:0     1                        (FILLER1 : 0..2048; [FOR DOWNWARD COMPATIBILITY,13 BITS]
 98   0    1:0     1                         DVID: VID;                (*NAME OF DISK VOLUME*)
 99   0    1:0     1                         DEOVBLK: INTEGER;         (*LASTBLK OF VOLUME*)
100   0    1:0     1                         DNUMFILES: DIRRANGE;      (*NUM FILES IN DIR*)
101   0    1:0     1                         DLOADTIME: INTEGER;       (*TIME OF LAST ACCESS*)
102   0    1:0     1                         DLASTBOOT: DATEREC);      (*MOST RECENT DATE SETTING*)
103   0    1:0     1                     XDSKFILE,CODEFILE,TEXTFILE,INFOFILE,
104   0    1:0     1                     DATAFILE,GRAFFILE,FOTOFILE:
105   0    1:0     1                        (FILLER2 : 0..1024; [FOR DOWNWARD COMPATIBILITY]
106   0    1:0     1                         STATUS : BOOLEAN;          [FOR FILER WILDCARDS]
107   0    1:0     1                         DTID: TID;                (*TITLE OF FILE*)
108   0    1:0     1                         DLASTBYTE: 1..FBLKSIZE;   (*NUM BYTES IN LAST BLOCK*)
109   0    1:0     1                         DACCESS: DATEREC)         (*LAST MODIFICATION DATE*)
110   0    1:0     1                  END (*DIRENTRY*) ;
111   0    1:0     1
112   0    1:0     1      DIRP = ^DIRECTORY;
113   0    1:0     1
114   0    1:0     1      DIRECTORY = ARRAY [DIRRANGE] OF DIRENTRY;
115   0    1:0     1
116   0    1:0     1                                             (*FILE INFORMATION*)
117   0    1:0     1
118   0    1:0     1      CLOSETYPE = (CNORMAL,CLOCK,CPURGE,CCRUNCH);
119   0    1:0     1      WINDOWP = ^WINDOW;
120   0    1:0     1      WINDOW = PACKED ARRAY [0..0] OF CHAR;
121   0    1:0     1      FIBP = ^FIB;
122   0    1:0     1
```

```
123   0     1:0      1      FIB = RECORD
124   0     1:0      1              FWINDOW: WINDOWP;  (*USER WINDOW...F^, USED BY GET-PUT*)
125   0     1:0      1              FEOF,FEOLN: BOOLEAN;
126   0     1:0      1              FSTATE: (FJANDW,FNEEDCHAR,FGOTCHAR);
127   0     1:0      1              FRECSIZE: INTEGER; (*IN BYTES...0=>BLOCKFILE, 1=>CHARFILE*)
128   0     1:0      1              CASE FISOPEN: BOOLEAN OF
129   0     1:0      1                TRUE: (FISBLKD: BOOLEAN; (*FILE IS ON BLOCK DEVICE*)
130   0     1:0      1                       FUNIT: UNITNUM;   (*PHYSICAL UNIT #*)
131   0     1:0      1                       FVID: VID;        (*VOLUME NAME*)
132   0     1:0      1                       FREPTCNT,         (* # TIMES F^ VALID W/O GET*)
133   0     1:0      1                       FNXTBLK,          (*NEXT REL BLOCK TO IO*)
134   0     1:0      1                       FMAXBLK: INTEGER; (*MAX REL BLOCK ACCESSED*)
135   0     1:0      1                       FMODIFIED:BOOLEAN;(*PLEASE SET NEW DATE IN CLOSE*)
136   0     1:0      1                       FHEADER: DIRENTRY;(*COPY OF DISK DIR ENTRY*)
137   0     1:0      1                       CASE FSOFTBUF: BOOLEAN OF (*DISK GET-PUT STUFF*)
138   0     1:0      1                         TRUE: (FNXTBYTE,FMAXBYTE: INTEGER;
139   0     1:0      1                                FBUFCHNGD: BOOLEAN;
140   0     1:0      1                                FBUFFER: PACKED ARRAY [0..FBLKSIZE] OF CHAR))
141   0     1:0      1          END (*FIB*) ;
142   0     1:0      1
143   0     1:0      1                                          (*USER WORKFILE STUFF*)
144   0     1:0      1
145   0     1:0      1      INFOREC = RECORD
146   0     1:0      1                SYMFIBP,CODEFIBP: FIBP;          (*WORKFILES FOR SCRATCH*)
147   0     1:0      1                ERRSYM,ERRBLK,ERRNUM: INTEGER;   (*ERROR STUFF IN EDIT*)
148   0     1:0      1                SLOWTERM,STUPID: BOOLEAN;        (*STUDENT PROGRAMMER ID!!*)
149   0     1:0      1                ALTMODE: CHAR;                   (*WASHOUT CHAR FOR COMPILER*)
150   0     1:0      1                GOTSYM,GOTCODE: BOOLEAN;         (*TITLES ARE MEANINGFUL*)
151   0     1:0      1                WORKVID,SYMVID,CODEVID: VID;     (*PERM&CUR WORKFILE VOLUMES*)
152   0     1:0      1                WORKTID,SYMTID,CODETID: TID      (*PERM&CUR WORKFILES TITLE*)
153   0     1:0      1              END (*INFOREC*) ;
154   0     1:0      1
155   0     1:0      1                                          (*CODE SEGMENT LAYOUTS*)
156   0     1:0      1
157   0     1:0      1      SEGRANGE = 0..MAXSEG;
158   0     1:0      1      SEGDESC = RECORD
159   0     1:0      1                DISKADDR: INTEGER;       (*REL BLK IN CODE...ABS IN SYSCOM^*)
160   0     1:0      1                CODELENG: INTEGER        (*# BYTES TO READ IN*)
161   0     1:0      1              END (*SEGDESC*) ;
162   0     1:0      1
163   0     1:0      1                                          (*DEBUGGER STUFF*)
```

```
164   0     1:0     1
165   0     1:0     1     BYTERANGE = 0..255;
166   0     1:0     1     TRICKARRAY = RECORD          [MEMORY DIDDLING FOR EXECERROR]
167   0     1:0     1                    CASE BOOLEAN OF
168   0     1:0     1                      TRUE : (WORD : ARRAY [0..0] OF INTEGER);
169   0     1:0     1                      FALSE : (BYTE : PACKED ARRAY [0..0] OF BYTERANGE)
170   0     1:0     1                    END;
171   0     1:0     1     MSCWP = ^ MSCW;              (*MARK STACK RECORD POINTER*)
172   0     1:0     1     MSCW = RECORD
173   0     1:0     1              STATLINK: MSCWP;   (*POINTER TO PARENT MSCW*)
174   0     1:0     1              DYNLINK: MSCWP;    (*POINTER TO CALLER'S MSCW*)
175   0     1:0     1              MSSEG,MSJTAB: ^TRICKARRAY;
176   0     1:0     1              MSIPC: INTEGER;
177   0     1:0     1              LOCALDATA: TRICKARRAY
178   0     1:0     1            END (*MSCW*) ;
179   0     1:0     1
180   0     1:0     1                                          (*SYSTEM COMMUNICATION AREA*)
181   0     1:0     1                                          (*SEE INTERPRETERS...NOTE  *)
182   0     1:0     1                                          (*THAT WE ASSUME BACKWARD  *)
183   0     1:0     1                                          (*FIELD ALLOCATION IS DONE *)
184   0     1:0     1     SYSCOMREC = RECORD
185   0     1:0     1                   IORSLT: IORSLTWD;     (*RESULT OF LAST IO CALL*)
186   0     1:0     1                   XEQERR: INTEGER;      (*REASON FOR EXECERROR CALL*)
187   0     1:0     1                   SYSUNIT: UNITNUM;     (*PHYSICAL UNIT OF BOOTLOAD*)
188   0     1:0     1                   BUGSTATE: INTEGER;    (*DEBUGGER INFO*)
189   0     1:0     1                   GDIRP: DIRP;          (*GLOBAL DIR POINTER,SEE VOLSEARCH*)
190   0     1:0     1                   LASTMP,STKBASE,BOMBP: MSCWP;
191   0     1:0     1                   MEMTOP,SEG,JTAB: INTEGER;
192   0     1:0     1                   BOMBIPC: INTEGER;     (*WHERE XEQERR BLOWUP WAS*)
193   0     1:0     1                   HLTLINE: INTEGER;     (*MORE DEBUGGER STUFF*)
194   0     1:0     1                   BRKPTS: ARRAY [0..3] OF INTEGER;
195   0     1:0     1                   RETRIES: INTEGER;     (*DRIVERS PUT RETRY COUNTS*)
196   0     1:0     1                   EXPANSION: ARRAY [0..8] OF INTEGER;
197   0     1:0     1                   HIGHTIME,LOWTIME: INTEGER;
198   0     1:0     1                   MISCINFO: PACKED RECORD
199   0     1:0     1                               NOBREAK,STUPID,SLOWTERM,
200   0     1:0     1                               HASXYCRT,HASLCCRT,HAS8510A,HASCLOCK: BOOLEAN;
201   0     1:0     1                               USERKIND:(NORMAL, AQUIZ, BOOKER, PQUIZ);
202   0     1:0     1                               IS_FLIPT : BOOLEAN
203   0     1:0     1                             END;
204   0     1:0     1                   CRTTYPE: INTEGER;
```

```
205   0    1:0     1                    CRTCTRL: PACKED RECORD
206   0    1:0     1                               RLF,NDFS,ERASEEOL,ERASEEOS,HOME,ESCAPE: CHAR;
207   0    1:0     1                               BACKSPACE: CHAR;
208   0    1:0     1                               FILLCOUNT: 0..255;
209   0    1:0     1                               CLEARSCREEN, CLEARLINE: CHAR;
210   0    1:0     1                               PREFIXED: PACKED ARRAY [0..8] OF BOOLEAN
211   0    1:0     1                             END;
212   0    1:0     1                    CRTINFO: PACKED RECORD
213   0    1:0     1                               WIDTH,HEIGHT: INTEGER;
214   0    1:0     1                               RIGHT,LEFT,DOWN,UP: CHAR;
215   0    1:0     1                               BADCH,CHARDEL,STOP,BREAK,FLUSH,EOF: CHAR;
216   0    1:0     1                               ALTMODE,LINEDEL: CHAR;
217   0    1:0     1                               BACKSPACE,ETX,PREFIX: CHAR;
218   0    1:0     1                               PREFIXED: PACKED ARRAY [0..13] OF BOOLEAN
219   0    1:0     1                             END;
220   0    1:0     1                    SEGTABLE: ARRAY [SEGRANGE] OF
221   0    1:0     1                                RECORD
222   0    1:0     1                                  CODEUNIT: UNITNUM;
223   0    1:0     1                                  CODEDESC: SEGDESC
224   0    1:0     1                                END
225   0    1:0     1                  END (*SYSCOM*);
226   0    1:0     1
227   0    1:0     1      MISCINFOREC = RECORD
228   0    1:0     1                      MSYSCOM: SYSCOMREC
229   0    1:0     1                    END;
230   0    1:0     1
231   0    1:0     1  VAR
232   0    1:0     1      SYSCOM: ^SYSCOMREC;                    (*MAGIC PARAM...SET UP IN BOOT*)
233   0    1:0     2      GFILES: ARRAY [0..5] OF FIBP;          (*GLOBAL FILES, 0=INPUT, 1=OUTPUT*)
234   0    1:0     8      USERINFO: INFOREC;                     (*WORK STUFF FOR COMPILER ETC*)
235   0    1:0    54      EMPTYHEAP: ^INTEGER;                   (*HEAP MARK FOR MEM MANAGING*)
236   0    1:0    55      INPUTFIB,OUTPUTFIB,                    (*CONSOLE FILES...GFILES ARE COPIES*)
237   0    1:0    55      SYSTERM,SWAPFIB: FIBP;                 (*CONTROL AND SWAPSPACE FILES*)
238   0    1:0    59      SYVID,DKVID: VID;                      (*SYSUNIT VOLID & DEFAULT VOLID*)
239   0    1:0    67      THEDATE: DATEREC;                      (*TODAY...SET IN FILER OR SIGN ON*)
240   0    1:0    68      DEBUGINFO: ^INTEGER;                   (*DEBUGGERS GLOBAL INFO WHILE RUNIN*)
241   0    1:0    69      STATE: CMDSTATE;                       (*FOR GETCOMMAND*)
242   0    1:0    70      PL: STRING;                            (*PROMPTLINE STRING...SEE PROMPT*)
243   0    1:0   111      IPOT: ARRAY [0..4] OF INTEGER;         (*INTEGER POWERS OF TEN*)
244   0    1:0   116      FILLER: STRING[FILL_LEN];              (*NULLS FOR CARRIAGE DELAY*)
245   0    1:0   122      DIGITS: SET OF '0'..'9';
```

```
246   0    1:0   126     UNITABLE: ARRAY [UNITNUM] OF (*0 NOT USED*)
247   0    1:0   126                 RECORD
248   0    1:0   126                   UVID: VID;     (*VOLUME ID FOR UNIT*)
249   0    1:0   126                   CASE UISBLKD: BOOLEAN OF
250   0    1:0   126                     TRUE: (UEOVBLK: INTEGER)
251   0    1:0   126                 END (*UNITABLE*) ;
252   0    1:0   264     FILENAME : FILE_TABLE;
253   0    1:0   264
254   0    1:0   264 (*-------------------------------------------------------------------------------*)
255   0    1:0   264 (* SYSTEM PROCEDURE FORWARD DECLARATIONS *)
256   0    1:0   264 (* THESE ARE ADDRESSED BY OBJECT CODE... *)
257   0    1:0   264 (*  DO NOT MOVE WITHOUT CAREFUL THOUGHT  *)
258   0    1:0   264
259   0    2:0     1 PROCEDURE EXECERROR;
260   0    2:0     1   FORWARD;
261   0    3:0     1 PROCEDURE FINIT(VAR F: FIB; WINDOW: WINDOWP; RECWORDS: INTEGER);
262   0    3:0     4   FORWARD;
263   0    4:0     1 PROCEDURE FRESET(VAR F: FIB);
264   0    4:0     2   FORWARD;
265   0    5:0     1 PROCEDURE FOPEN(VAR F: FIB; VAR FTITLE: STRING;
266   0    5:0     3                   FOPENOLD: BOOLEAN; JUNK: FIBP);
267   0    5:0     5   FORWARD;
268   0    6:0     1 PROCEDURE FCLOSE(VAR F: FIB; FTYPE: CLOSETYPE);
269   0    6:0     3   FORWARD;
270   0    7:0     1 PROCEDURE FGET(VAR F: FIB);
271   0    7:0     2   FORWARD;
272   0    8:0     1 PROCEDURE FPUT(VAR F: FIB);
273   0    8:0     2   FORWARD;
274   0    9:0     1 PROCEDURE XSEEK;
275   0    9:0     1   FORWARD;
276   0   10:0     3 FUNCTION FEOF(VAR F: FIB): BOOLEAN;
277   0   10:0     4   FORWARD;
278   0   11:0     3 FUNCTION FEOLN(VAR F: FIB): BOOLEAN;
279   0   11:0     4   FORWARD;
280   0   12:0     1 PROCEDURE FREADINT(VAR F: FIB; VAR I: INTEGER);
281   0   12:0     3   FORWARD;
282   0   13:0     1 PROCEDURE FWRITEINT(VAR F: FIB; I,RLENG: INTEGER);
283   0   13:0     4   FORWARD;
284   0   14:0     1 PROCEDURE XREADREAL;
285   0   14:0     1   FORWARD;
286   0   15:0     1 PROCEDURE XWRITEREAL;
```

```
287   0   15:0    1   FORWARD;
288   0   16:0    1 PROCEDURE FREADCHAR(VAR F: FIB; VAR CH: CHAR);
289   0   16:0    3   FORWARD;
290   0   17:0    1 PROCEDURE FWRITECHAR(VAR F: FIB; CH: CHAR; RLENG: INTEGER);
291   0   17:0    4   FORWARD;
292   0   18:0    1 PROCEDURE FREADSTRING(VAR F: FIB; VAR S: STRING; SLENG: INTEGER);
293   0   18:0    4   FORWARD;
294   0   19:0    1 PROCEDURE FWRITESTRING(VAR F: FIB; VAR S: STRING; RLENG: INTEGER);
295   0   19:0    4   FORWARD;
296   0   20:0    1 PROCEDURE FWRITEBYTES(VAR F: FIB; VAR A: WINDOW; RLENG,ALENG: INTEGER);
297   0   20:0    5   FORWARD;
298   0   21:0    1 PROCEDURE FREADLN(VAR F: FIB);
299   0   21:0    2   FORWARD;
300   0   22:0    1 PROCEDURE FWRITELN(VAR F: FIB);
301   0   22:0    2   FORWARD;
302   0   23:0    1 PROCEDURE SCONCAT(VAR DEST,SRC: STRING; DESTLENG: INTEGER);
303   0   23:0    4   FORWARD;
304   0   24:0    1 PROCEDURE SINSERT(VAR SRC,DEST: STRING; DESTLENG,INSINX: INTEGER);
305   0   24:0    5   FORWARD;
306   0   25:0    1 PROCEDURE SCOPY(VAR SRC,DEST: STRING; SRCINX,COPYLENG: INTEGER);
307   0   25:0    5   FORWARD;
308   0   26:0    1 PROCEDURE SDELETE(VAR DEST: STRING; DELINX,DELLENG: INTEGER);
309   0   26:0    4   FORWARD;
310   0   27:0    3 FUNCTION SPOS(VAR TARGET,SRC: STRING): INTEGER;
311   0   27:0    5   FORWARD;
312   0   28:0    3 FUNCTION FBLOCKIO(VAR F: FIB; VAR A: WINDOW; I: INTEGER;
313   0   28:0    6                   NBLOCKS,RBLOCK: INTEGER; DOREAD: BOOLEAN): INTEGER;
314   0   28:0    9   FORWARD;
315   0   29:0    1 PROCEDURE FGOTOXY(X,Y: INTEGER);
316   0   29:0    3   FORWARD;
317   0   29:0    3
318   0   29:0    3 (* NON FIXED FORWARD DECLARATIONS *)
319   0   29:0    3
320   0   30:0    3 FUNCTION VOLSEARCH(VAR FVID: VID; LOOKHARD: BOOLEAN;
321   0   30:0    5                    VAR FDIR: DIRP): UNITNUM;
322   0   30:0    6   FORWARD;
323   0   31:0    1 PROCEDURE WRITEDIR(FUNIT: UNITNUM; FDIR: DIRP);
324   0   31:0    3   FORWARD;
325   0   32:0    3 FUNCTION DIRSEARCH(VAR FTID: TID; FINDPERM: BOOLEAN; FDIR: DIRP): DIRRANGE;
326   0   32:0    6   FORWARD;
327   0   33:0    3 FUNCTION SCANTITLE(FTITLE: STRING; VAR FVID: VID; VAR FTID: TID;
```

```
328   0    33:0      6                          VAR FSEGS: INTEGER; VAR FKIND: FILEKIND): BOOLEAN;
329   0    33:0     49    FORWARD;
330   0    34:0      1  PROCEDURE DELENTRY(FILX: DIRRANGE; FDIR: DIRP);
331   0    34:0      3    FORWARD;
332   0    35:0      1  PROCEDURE INSENTRY(VAR FENTRY: DIRENTRY; FINX: DIRRANGE; FDIR: DIRP);
333   0    35:0      4    FORWARD;
334   0    36:0      1  PROCEDURE HOMECURSOR;
335   0    36:0      1    FORWARD;
336   0    37:0      1  PROCEDURE CLEARSCREEN;
337   0    37:0      1    FORWARD;
338   0    38:0      1  PROCEDURE CLEARLINE;
339   0    38:0      1    FORWARD;
340   0    39:0      1  PROCEDURE PROMPT;
341   0    39:0      1    FORWARD;
342   0    40:0      3  FUNCTION SPACEWAIT(FLUSH: BOOLEAN): BOOLEAN;
343   0    40:0      4    FORWARD;
344   0    41:0      3  FUNCTION GETCHAR(FLUSH: BOOLEAN): CHAR;
345   0    41:0      4    FORWARD;
346   0    42:0      3  FUNCTION FETCHDIR(FUNIT:UNITNUM) : BOOLEAN;
347   0    42:0      4    FORWARD;
348   0    43:0      1  PROCEDURE COMMAND;
349   0    43:0      1    FORWARD;
350   0    43:0      1
351   0    43:0      1
352   0    43:0      1  (*$I GLOBALS.TEXT*)
353   0    43:0      1
354   0    43:0      1  (*-----------------------------------------------------------*)
355   0    43:0      1
356   1     1:0      1  SEPARATE UNIT PASCALIO;
357   1     1:0      1  INTERFACE
358   1     1:0      1
359   1     1:0      1    TYPE DECMAX = INTEGER[36];
360   1     1:0      1         STUNT = RECORD CASE INTEGER OF
361   1     1:0      1                   2:(W2:INTEGER[4]);
362   1     1:0      1                   3:(W3:INTEGER[8]);
363   1     1:0      1                   4:(W4:INTEGER[12]);
364   1     1:0      1                   5:(W5:INTEGER[16]);
365   1     1:0      1                   6:(W6:INTEGER[20]);
366   1     1:0      1                   7:(W7:INTEGER[24]);
367   1     1:0      1                   8:(W8:INTEGER[28]);
368   1     1:0      1                   9:(W9:INTEGER[32]);
```

```
369   1    1:0     1                10:(w10:INTEGER[36])
370   1    1:0     1              END;
371   1    1:0     1
372   1    1:0     1
373   1    1:0     1  PROCEDURE FSEEK(VAR F: FIB; RECNUM: INTEGER);
374   1    2:0     1  PROCEDURE FREADREAL(VAR F: FIB; VAR X: REAL);
375   1    3:0     1  PROCEDURE FWRITEREAL(VAR F: FIB; X: REAL; W, D: INTEGER);
376   1    4:0     1  PROCEDURE FREADDEC(VAR F: FIB; VAR D: STUNT; L: INTEGER);
377   1    5:0     1  PROCEDURE FWRITEDEC(VAR F: FIB; D: DECMAX; RLENG: INTEGER);
378   1    6:0     3  FUNCTION SUPER_MOD(A,B : INTEGER) : INTEGER;
379   1    7:0     3  FUNCTION SUPER_DIV(A,B : INTEGER) : INTEGER;
380   1    7:0     5
381   1    1:0     5 IMPLEMENTATION
382   1    1:0     1
383   1    6:0     3  FUNCTION SUPER_MOD[A,B : INTEGER) : INTEGER];
384   1    6:0     5                  [CALCULATES A*B MOD 512 WITH 0 <= A,B <= MAXINT]
385   1    6:0     5  VAR  TEMP1, TEMP2, TEMP3 : INTEGER;
386   1    6:0     0  BEGIN
387   1    6:1     0    TEMP1 := (A MOD 1024 DIV 32) * (B MOD 32) * 32 MOD 512;
388   1    6:1    19    TEMP2 := (A MOD 32) * (B MOD 1024 DIV 32) * 32 MOD 512;
389   1    6:1    38    TEMP3 := (A MOD 32) * (B MOD 32) MOD 512;
390   1    6:1    51    SUPER_MOD := (TEMP1 + TEMP2 + TEMP3) MOD 512;
391   1    6:0    62    END   [OF SUPER_MOD];
392   1    6:0    74
393   1    6:0    74
394   1    8:0     3  FUNCTION LITTLE_DIV(A,B,C : INTEGER) : INTEGER;
395   1    8:0     0  BEGIN
396   1    8:1     0    LITTLE_DIV := (A MOD 512 + B MOD 512 + C MOD 512) DIV 512;
397   1    8:0    23    END  [OF LITTLE_DIV];
398   1    8:0    36
399   1    7:0     3  FUNCTION SUPER_DIV[A,B : INTEGER) : INTEGER];
400   1    7:0     5                  [CALCULATES A*B DIV 512 WITH 0 <= A*B <= 2**24]
401   1    7:0     5  VAR  A_HI, A_MID, A_LOW, B_HI, B_MID, B_LOW : INTEGER;
402   1    7:0     0  BEGIN          [OF SUPER_DIV]
403   1    7:1     0    A_HI := A DIV 1024;
404   1    7:1     7    A_MID := A MOD 1024 DIV 32;
405   1    7:1    16    A_LOW := A MOD 32;
406   1    7:1    21    B_HI := B DIV 1024;
407   1    7:1    28    B_MID := B MOD 1024 DIV 32;
408   1    7:1    37    B_LOW := B MOD 32;
409   1    7:1    42    SUPER_DIV := A_HI * B_HI * 2048 + A_HI * B_MID * 64 + A_HI * B_LOW * 2
```

```
410   1    7:1    98                + A_MID * B_HI * 64 + A_MID * B_MID * 2 + A_MID * B_LOW DIV 16
411   1    7:1    76                + A_LOW * B_HI * 2 + A_LOW * B_MID DIV 16 + A_LOW * B_LOW DIV 512
412   1    7:1    94                + LITTLE_DIV(A_MID * B_LOW * 32,A_LOW * B_MID * 32,A_LOW * B_LOW);
413   1    7:3   119      END  [OF SUPER_DIV];
414   1    7:0   132
415   1    1:0     1    PROCEDURE FSEEK(*VAR F: FIB; RECNUM: INTEGER*);
416   1    1:0     3      LABEL 1;
417   1    1:0     3      VAR BYTE,BLOCK,N: INTEGER;
418   1    1:0     0    BEGIN SYSCOM^.IORSLT := INOERROR;
419   1    1:1     5      IF F.FISOPEN THEN
420   1    1:2     9        WITH F,FHEADER DO
421   1    1:3    17          BEGIN
422   1    1:4    17            IF (RECNUM < 0) OR NOT FSOFTBUF OR
423   1    1:4    25                    ((DFKIND = TEXTFILE) AND (FRECSIZE = 1)) THEN
424   1    1:5    41              GOTO 1; (*NO SEEK ALLOWED*)
425   1    1:4    43            BLOCK := SUPERDIV(RECNUM,FRECSIZE) + 1;
426   1    1:4    54                  [RECNUM*FRECSIZE DIV FBLKSIZE + 1;]
427   1    1:4    54            BYTE := SUPERMOD(RECNUM,FRECSIZE);
428   1    1:4    63                  [BYTE := RECNUM*FRECSIZE MOD FBLKSIZE;]
429   1    1:4    63            IF BYTE = 0 THEN
430   1    1:5    68              BEGIN
431   1    1:6    68                BYTE := FBLKSIZE;
432   1    1:6    73                BLOCK := BLOCK - 1;
433   1    1:5    78                END;
434   1    1:4    78            N := DLASTBLK-DFIRSTBLK;
435   1    1:4    85            IF (BLOCK > N) OR ((BLOCK = N) AND (BYTE >= DLASTBYTE)) THEN
436   1    1:5   100              BEGIN BLOCK := N; BYTE := DLASTBYTE END;
437   1    1:4   108            IF BLOCK <> FNXTBLK THEN
438   1    1:5   115              BEGIN
439   1    1:6   115                IF FBUFCHNGD THEN
440   1    1:7   120                  BEGIN FBUFCHNGD := FALSE; FMODIFIED := TRUE;
441   1    1:8   130                    UNITWRITE(FUNIT,FBUFFER,FBLKSIZE,DFIRSTBLK+FNXTBLK-1);
442   1    1:8   150                    IF IORESULT <> ORD(INOERROR) THEN GOTO 1
443   1    1:7   158                  END;
444   1    1:6   158                IF (BLOCK <= FMAXBLK) AND (BYTE <> FBLKSIZE) THEN
445   1    1:7   171                  BEGIN
446   1    1:8   171                    UNITREAD(FUNIT,FBUFFER,FBLKSIZE,DFIRSTBLK+BLOCK-1);
447   1    1:8   189                    IF IORESULT <> ORD(INOERROR) THEN GOTO 1
448   1    1:7   197                  END
449   1    1:5   197              END;
450   1    1:4   197            IF FNXTBLK > FMAXBLK THEN
```

```
451   1    1:5   206            BEGIN FMAXBLK := FNXTBLK; FMAXBYTE := FNXTBYTE END
452   1    1:4   220          ELSE
453   1    1:5   222            IF (FNXTBLK = FMAXBLK) AND (FNXTBYTE > FMAXBYTE) THEN
454   1    1:6   239              FMAXBYTE := FNXTBYTE;
455   1    1:4   246          FEOF := FALSE; FEOLN := FALSE; FREPTCNT := 0;
456   1    1:4   261          IF FSTATE <> FJANDW THEN FSTATE := FNEEDCHAR;
457   1    1:4   272          FNXTBLK := BLOCK; FNXTBYTE := BYTE
458   1    1:3   290        END
459   1    1:1   282    ELSE SYSCOM^.IORSLT := INOTOPEN;
460   1    1:1   289  1:
461   1    1:0   289  END (*FSEEK*) ;
462   1    1:0   306
463   1    2:0     1  PROCEDURE FREADREAL(*VAR F: FIB; VAR X: REAL*);
464   1    2:0     3    LABEL 1;
465   1    2:0     3    VAR CH: CHAR; NEG,XVALID: BOOLEAN; IPOT: INTEGER;
466   1    2:0     0  BEGIN
467   1    2:1     0    WITH F DO
468   1    2:2     3        BEGIN X := 0; NEG := FALSE; XVALID := FALSE;
469   1    2:3    14          IF FSTATE = FNEEDCHAR THEN FGET(F);
470   1    2:3    24          WHILE (FWINDOW^[0] = ' ') AND NOT FEOF DO FGET(F);
471   1    2:3    42          IF FEOF THEN GOTO 1;
472   1    2:3    48          CH := FWINDOW^[0];
473   1    2:3    54          IF (CH = '+') OR (CH = '-') THEN
474   1    2:4    63            BEGIN NEG := CH = '-'; FGET(F); CH := FWINDOW^[0] END;
475   1    2:3    78          WHILE (CH IN DIGITS) AND NOT FEOF DO
476   1    2:4    92            BEGIN XVALID := TRUE;
477   1    2:5    95              X := X*10 + (ORD(CH)-ORD('0'));
478   1    2:5   109              FGET(F); CH := FWINDOW^[0]
479   1    2:4   116            END;
480   1    2:3   121          IF FEOF THEN GOTO 1;
481   1    2:3   127          IPOT := -1;
482   1    2:3   131          IF CH = '.' THEN
483   1    2:4   136            BEGIN IPOT := 0;
484   1    2:5   139              REPEAT FGET(F); CH := FWINDOW^[0];
485   1    2:6   149                IF CH IN DIGITS THEN
486   1    2:7   159                  BEGIN XVALID := TRUE; IPOT := IPOT + 1;
487   1    2:8   167                    X := X + (ORD(CH)-ORD('0'))/PWROFTEN(IPOT)
488   1    2:7   175                  END
489   1    2:5   182              UNTIL FEOF OR NOT (CH IN DIGITS);
490   1    2:5   196              IF FEOF THEN GOTO 1
491   1    2:4   202            END;
```

```
492   1    2:3   232          IF ((CH = 'E') OR (CH = 'E')) AND (XVALID OR (IPOT < 0)) THEN
493   1    2:4   217            BEGIN
494   1    2:5   217              IF FSTATE = FJANDW THEN FGET(F)
495   1    2:5   224              ELSE FSTATE := FNEEDCHAR;
496   1    2:5   234              FREADINT(F,IPOT);
497   1    2:5   240              IF FEOF THEN GOTO 1;
498   1    2:5   246              IF NOT XVALID THEN X := 1; XVALID := TRUE;
499   1    2:5   253              IF IPOT < 0 THEN X := X/PWROFTEN(ABS(IPOT))
500   1    2:5   269              ELSE X := X*PWROFTEN(IPOT)
501   1    2:4   281            END;
502   1    2:3   286          IF XVALID THEN
503   1    2:4   289            IF NEG THEN X := -X
504   1    2:4   293            ELSE
505   1    2:3   301          ELSE SYSCOM^.IORSLT := IBADFORMAT
506   1    2:2   306        END;
507   1    2:1   308  1:
508   1    2:0   308  END (*FREADREAL*) ;
509   1    2:0   328
510   1    3:0     1  PROCEDURE FWRITEREAL(*X:REAL;  W, D: INTEGER*);
511   1    3:0     6  VAR J, TRUNCX, EXPX: INTEGER;
512   1    3:0     9      NORMX: REAL;  S: STRING[30];
513   1    3:0    27
514   1    3:0     0  BEGIN
515   1    3:0     0  (* CHECK W AND D FOR VALIDITY *)
516   1    3:1     0  IF (W < 0) OR (D < 0) THEN  BEGIN W := 0;  D := 0 END;
517   1    3:1    15
518   1    3:1    15  (* TAKE ABS(X), NORMALIZE IT AND CALCULATE EXPONENT *)
519   1    3:1    15  IF X < 0 THEN  BEGIN X := -X;  S[1] := '-' END
520   1    3:1    39            ELSE  S[1] := ' ';
521   1    3:1    46  EXPX := 0;  NORMX := X;
522   1    3:1    57  IF X >= PWROFTEN(0) THEN  (* DIVIDE DOWN TO SIZE *)
523   1    3:2    68    WHILE NORMX >= PWROFTEN(1) DO
524   1    3:3    79      BEGIN EXPX := EXPX+1;  NORMX := X/PWROFTEN(EXPX) END
525   1    3:1    96  ELSE
526   1    3:2   100    IF X <> 0 THEN  (* MULTIPLY UP TO SIZE *)
527   1    3:3   110      REPEAT
528   1    3:4   110        EXPX := EXPX-1;  NORMX := X*PWROFTEN(-EXPX)
529   1    3:3   123      UNTIL NORMX >= PWROFTEN(0);
530   1    3:3   139
531   1    3:3   139  (* ROUND NUMBER ACCORDING TO SOME VERY TRICKY RULES *)
532   1    3:1   139  IF (D=0) OR (D+EXPX+1 > 6) THEN  (* SCIENTIFIC NOTATION, OR DECIMAL PLACES *)
```

```
533   1     3:2   152     NORMX := NORMX + 5/PWROFTEN(6)          (* OVERSPECIFIED *)
534   1     3:1   160   ELSE IF D+EXPX+1 >= 0 THEN
535   1     3:3   178     NORMX := NORMX + 5/PWROFTEN(D+EXPX+1);
536   1     3:3   197   (* IF D+EXPX+1 < 0, THEN NUMBER IS EFFECTIVELY 0.0 *)
537   1     3:3   197
538   1     3:3   197   (* IF WE JUST BLEW NORMALIZED STUFF THEN FIX IT UP *)
539   1     3:1   197   IF NORMX >= PWROFTEN(1) THEN
540   1     3:2   208     BEGIN  EXPX := EXPX+1;  NORMX := NORMX/PWROFTEN(1) END;
541   1     3:2   225
542   1     3:2   225   (* PUT THE DIGITS INTO A STRING *)
543   1     3:1   225   FOR J := 3 TO 8 DO
544   1     3:2   237     BEGIN
545   1     3:3   237       TRUNCX := TRUNC(NORMX);
546   1     3:3   245       S[J] := CHR(TRUNCX+ORD('0'));
547   1     3:3   252       NORMX := (NORMX-TRUNCX)*PWROFTEN(1)
548   1     3:2   262     END;
549   1     3:2   274
550   1     3:2   274   (* PUT NUMBER INTO PROPER FORM *)
551   1     3:1   274   IF (D=0) OR (EXPX >= 6) THEN  (* SCIENTIFIC NOTATION *)
552   1     3:2   283     BEGIN
553   1     3:3   283       S[2] := S[3];
554   1     3:3   291       S[3] := '.';
555   1     3:3   296       J := 8;
556   1     3:3   299       IF EXPX <> 0 THEN
557   1     3:4   304         BEGIN
558   1     3:5   304           J := 9;
559   1     3:5   307           S[9] := 'E';
560   1     3:5   312           IF EXPX < 0 THEN
561   1     3:6   317             BEGIN J := 10;  S[10] := '-';  EXPX := -EXPX END;
562   1     3:5   329           IF EXPX > 9 THEN
563   1     3:6   334             BEGIN
564   1     3:7   334               J := J+1;
565   1     3:7   339               S[J] := CHR(EXPX DIV 10 + ORD('0'));
566   1     3:6   348             END;
567   1     3:5   348           J := J+1;
568   1     3:5   353           S[J] := CHR(EXPX MOD 10 + ORD('0'))
569   1     3:4   361         END;
570   1     3:3   362       S[0] := CHR(J);
571   1     3:2   367     END
572   1     3:1   367   ELSE  (* SOME KIND OF FIXED POINT NOTATION *)
573   1     3:2   369     IF EXPX >= 0 THEN
```

```
574   1     3:3    374        BEGIN
575   1     3:4    374          MOVELEFT(S[3], S[2], EXPX+1);
576   1     3:4    385          S[3+EXPX] := '.';
577   1     3:4    392          FILLCHAR(S[9], D-(5-EXPX), ' '); (* BLANK FILL AT END IF PRECISION *)
578   1     3:4    403          S[0] := CHR(3+D+EXPX);           (* WAS OVER-SPECIFIED *)
579   1     3:3    412        END
580   1     3:2    412      ELSE
581   1     3:3    414        BEGIN
582   1     3:4    414          MOVERIGHT(S[3], S[3-EXPX], 6);  (* MAKE ROOM FOR LEADING ZEROES *)
583   1     3:4    425          S[2] := '0';
584   1     3:4    430          S[3] := '.';
585   1     3:4    435          FILLCHAR(S[4], -EXPX-1, '0');  (* PUT IN LEADING ZEROES *)
586   1     3:4    445          FILLCHAR(S[9-EXPX], D-6+EXPX, ' ');(* PUT IN BLANKS FOR OVER-PRECISION*)
587   1     3:4    458          S[0] := CHR(3+D);
588   1     3:3    465        END;
589   1     3:1    465    IF W < LENGTH(S) THEN W := LENGTH(S);
590   1     3:1    479    FWRITESTRING( F, S, W );
591   1     3:0    486    END;  (*PROCEDURE WRITE_REAL *)
592   1     3:0    504
593   1     5:D      1    PROCEDURE FWRITEDEC(*VAR F: FIB; D: DECMAX; RLENG: INTEGER*);
594   1     5:D     13    VAR S: STRING[38]; I: INTEGER;
595   1     5:0      0    BEGIN
596   1     5:1      0      STR(D,S);
597   1     5:1     11      FWRITESTRING(F,S,RLENG)
598   1     5:0     15    END (*FWRITEDEC*) ;
599   1     5:0     30
600   1     5:0     30
601   1     4:D      1    PROCEDURE FREADDEC(*VAR F:FIB; VAR D: STUNT; L: INTEGER*);
602   1     4:D      4      LABEL 1;
603   1     4:D      4      VAR CH: CHAR;
604   1     4:D      5          NEG,DVALID: BOOLEAN; DIG,I: INTEGER;
605   1     4:0      0    BEGIN
606   1     4:1      0      WITH F DO
607   1     4:2      3          BEGIN
608   1     4:3      3            WITH D DO
609   1     4:4      6              CASE L OF
610   1     4:4      9                2: W2 := 0;      3: W3 := 0;       4: W4 := 0;
611   1     4:4     48                5: W5 := 0;      6: W6 := 0;       7: W7 := 0;
612   1     4:4     87                8: W8 := 0;      9: W9 := 0;      10: W10 := 0
613   1     4:4    114              END;
614   1     4:3    152            NEG := FALSE; DVALID := FALSE;
```

```
615   1     4:3    153            IF FSTATE = FNEEDCHAR THEN FGET(F);
616   1     4:3    169            WHILE (FWINDOW^[0] = ' ') AND NOT FEOF DO FGET(F);
617   1     4:3    186            IF FEOF THEN GOTO 1;
618   1     4:3    192            CH := FWINDOW^[0];
619   1     4:3    198            IF (CH = '+') OR (CH = '-') THEN
620   1     4:4    207              BEGIN NEG := CH = '-'; FGET(F); CH := FWINDOW^[0] END;
621   1     4:3    222            WHILE (CH IN DIGITS) AND NOT FEOF DO
622   1     4:4    236              BEGIN DVALID := TRUE;
623   1     4:5    239                DIG:=ORD(CH)-ORD('0');
624   1     4:5    244                IF NEG THEN DIG:=-DIG;
625   1     4:5    251                WITH J DO
626   1     4:6    254                  CASE L OF
627   1     4:6    257                    2:W2:=10*W2+DIG;
628   1     4:6    284                    3:W3:=10*W3+DIG;
629   1     4:6    311                    4:W4:=10*W4+DIG;
630   1     4:6    338                    5:W5:=10*W5+DIG;
631   1     4:6    365                    6:W6:=10*W6+DIG;
632   1     4:6    392                    7:W7:=10*W7+DIG;
633   1     4:6    419                    8:W8:=10*W8+DIG;
634   1     4:6    446                    9:W9:=10*W9+DIG;
635   1     4:6    473                    10:W10:=10*W10+DIG;
636   1     4:6    500                  END;
637   1     4:5    526                FGET(F); CH := FWINDOW^[0]
638   1     4:4    533              END;
639   1     4:3    538            IF NOT (DVALID OR FEOF) THEN SYSCOM^.IORSLT := IBADFORMAT
640   1     4:2    548          END;
641   1     4:1    550    1:
642   1     4:0    550    END(*FREADDEC*) ;
643   1     4:0    576
644   1     4:0    576
645   1     1:0      0 END [ PASCALIO ] ;
646   1     1:0    480
647   1     1:0    480 (*DUMMY LEVEL 0 OUTERBLOCK*)
648   0     1:0      0 BEGIN END.
```

```
        ;.INCLUDE MACH.TYPE.TEXT

        .PROC   DECOPS
        ;
        ; COPYRIGHT (C) 1978,   THE REGENTS OF THE UNIVERSITY OF CALIFORNIA
        ;                       SAN DIEGO CAMPUS
        ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                              ;
;                          DECIMAL OPERATORS                                   ;
;                                                                              ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MP      .EQU    R5
IPC     .EQU    R4
BASE    .EQU    R3
BK      .EQU    R2

;.IF     SOBSXT=0
.MACRO  SOB             ; NOTE: THIS MACRO VERSION OF SOB DOES
        DEC     %1      ;    NOT(!) PRESERVE CONDITION CODES.
        BNE     %2
.ENDM

.MACRO  SXT             ;  THIS SXT MACRO DOES SUPPORT ALL
        BPL     $99     ;    ADDRESSING MODES.
        MOV     #-1,%1  ; THERE MUST BE A NON-LOCAL LABEL BETWEEN
        BR      $98     ;    ANY TWO SXTS IF LSI=0
$99:    CLR     %1
$98:
.ENDM
;.ENDC


        ; TRAP PARAMETERS
INTOVR  .EQU    5
DIVZER  .EQU    6
S2LONG  .EQU    15
```

```
        .DEF    GDEC

GDEC:   ; DECIMAL INSTRUCTION
        MOV     (SP)+,EXTRTN    ; SAVE RETURN ADDRESS
        MOV     MP,DECMP        ; SAVE REGISTERS
        MOV     IPC,DECIPC
        MOV     BASE,DECBAS
        MOV     BK,DECBK
        MOVB    (SP)+,R1        ; GRAB INSTRUCTION BYTE
        MOV     DECTBL(R1),PC   ; AND GO EXECUTE

BIGRTN: MOV     DECMP,MP        ; RESTORE REGISTERS
        MOV     DECIPC,IPC
        MOV     DECBAS,BASE
        MOV     DECBK,BK
        MOV     EXTRTN,PC       ; TRICKY RETURN TO CALLING ROUTINE

DECMP:  .WORD   0
DECIPC: .WORD   0
DECBAS: .WORD   0
DECBK:  .WORD   0
EXTRTN: .WORD   0

DECTBL: .WORD   DAJ
        .WORD   DAD
        .WORD   DSB
        .WORD   DNG
        .WORD   DMP
        .WORD   DDV
        .WORD   DSTR
        .WORD   DCV
        .WORD   DECCMP
        .WORD   DCVT
        .WORD   DTNC

;.IF SMLI=1
;MLI:    MUL     R5,R4
;       RTS     PC
;.ELSE
                ; SOFT MULTIPLY IF NO HARD MULTIP
MLI:    ; SOFT MULTIPLY   (R4,R5):=R5 X R4
```

```
        MOV     R0,SAV0         ; SAVE REGISTERS
        MOV     R1,SAV1
        CLR     -(SP)           ;  SIGN STORAGE
        TST     R5              ; CHECK MULTIPLICAND
        BGT     $1              ; SKIP FOLLOWING IF +
        BEQ     ZEROM           ; ANSWER IS ZERO
        INC     @SP             ; REMEMBER NEGATIVE
        NEG     R5
        BMI     SPECL1          ; SPECIAL  HANDLING FOR  -32768
$1:     TST     R4              ; TEST MULTIPLIER
        BGT     $2
        BEQ     ZEROM
        INC     @SP
        NEG     R4
        BMI     SPECL2
$2:     MOV     #16.,-(SP)      ; SET UP ITERATION COUNT
        CMP     R5,R4           ; MAKE SURE
        BGE     MCLR            ; MULTIPLIER
        MOV     R5,R0           ; IS
        MOV     R4,R5           ; SMALLER
        MOV     R0,R4
MCLR:   CLR     R0              ; CLEAR PRODUCT
        CLR     R1
MMUL:   ROR     R4              ; GET MULTIPLIER BIT
        BCC     $1              ;  = 0?
        ADD     R5,R1           ; NO, ADD IN MULTIPLICAND
        ADC     R0
        CLC
$1:     ROR     R0              ; ROTATE PRODUCT
        ROR     R1
        BCC     CYC
        BIS     #100000,R0
CYC:    DEC     @SP
        BGT     MMUL
        TST     (SP)+           ; GET RID OF COUNTER
        MOV     R0,R5           ; PUT RESULT IN OUTPUT REGISTERS
        MOV     R1,R4           ; NOTE REVERSAL OF REGISTERS
        ROR     (SP)+           ; DETERMINE SIGN
        BCC     OUTM            ; OF PRODUCT
        COM     R4
        NEG     R5
```

```
        BCS     $1
        INC     R4
$1:     BR      OUTM
SPECL2: MOV     R5,R4           ; R4 WAS -32768
                                ; ELSE R5 WAS -32768
SPECL1: CLR     R5
        ASR     (SP)+           ; WAS R5 NEGATED ALREADY?
        BNE     $1              ; YES
        NEG     R4              ; NO  NEGATE NOW
$1:     ASR     R4              ; DIVIDE BY 2
        BCC     OUTM
        ROR     R5
        TST     R4              ; FIX FOR NEGATIVE
        BPL     OUTM            ; ODD NUMBERS
        INC     R4
OUTM:   MOV     SAV0,R0          ; RESTORE REGISTERS
        MOV     SAV1,R1
        RTS     PC
ZEROM:  CLR     R4
        CLR     R5
        TST     (SP)+
        BR      OUTM
SAV0:   .WORD
SAV1:   .WORD
;.ENDC

DAJ:    ; DECIMAL ADJUST
        MOVB    (SP)+,R0        ; GET DESIRED LENGTH
        SUB     (SP)+,R0        ; TOSS OPERAND LEN; R0 = DIFF
        BEQ     DAJDNE
        BLT     SHRINK
        TST     @SP
        SXT     R1              ; SIGN EXTENSION
XPAND:  MOV     R1,-(SP)
        SOB     R0,XPAND
        BR      DAJDNE
SHRINK: NEG     R0
DLOOP:  TST     (SP)
        BEQ     DPOS
        INC     (SP)+
        BEQ     DNEG
```

```
HOLE:   JMP     DOVR            ; OVERFLOW
DNEG:   TST     (SP)
        BPL     HOLE            ; OVFL OCCURRED
        SOB     R0,DLOOP
        BR      DAJDNE
DPOS:   TST     (SP)+           ; KNOCK SP
        TST     (SP)
        BMI     HOLE
        SOB     R0,DLOOP
DAJDNE: JMP     @#BIGRTN        ; EXIT DECOPS

DCVT:   MOV     #1,-(SP)        ; PUSH LENGTH WORD OF 1
        JMP     @#BIGRTN

DTNC:   MOV     #1,-(SP)        ; PUSH DESIRED LENGTH OF 1
        BR      DAJ

DNG:    ; DECIMAL NEGATE
        MOV     SP,R1
        JSR     PC,DODNG
        JMP     @#BIGRTN          ; EXIT DECOPS


DODNG:  ; NEGATE SUBROUTINE.. BK IS DESTROYED
        MOV     (R1),BK         ; R1 POINTS TO LENGTH UPON ENTRY
        ASL     BK
        ADD     R1,BK           ; BK POINTS TO LSB
        MOV     (R1),R1         ; NOW R1 HAS LENGTH
        TST     (BK)+
        SEC
$1:     BCC     CRYCLR
        COM     -(BK)
        ADD     #1,(BK)
        SOB     R1,$1
        BVC     DNGEND
        TST     -(BK)
        TST     -(SP)           ; INSERT EXTRA WORD
        MOV     SP,R1
$2:     MOV     2(R1),(R1)+
        CMP     R1,BK
        BLO     $2
```

```
        TST     2(BK)
        SXT     (BK)
        BR      DNGEND
CRYCLR: COM     -(BK)
        SOB     R1,CRYCLR
DNGEND: MOV     BK,R1
        TST     -(R1)           ; RESTORE R1 TO ORIG. VALUE
        RTS     PC

DSB:    ; DECIMAL SUBTRACT
        MOV     SP,SUBFLAG      ; NONZERO VALUE INDICATES SUBTRACT
        JSR     PC,ADDSUB
        BR      DCH

DAD:    ; DECIMAL ADD
        CLR     SUBFLAG         ; ZERO INDICATOR FOR ADD
        JSR     PC,ADDSUB
        BR      DCH

ADDSUB: MOV     (SP)+,ASRET     ; SAVE RETURN ADDR
        MOV     (SP),R0         ; GET LENGTH
        ASL     R0              ; FOR BYTES
        ADD     SP,R0
        TST     (R0)+           ; POINT R0 TO 2ND OP LEN
        CMP     @R0,@SP         ; COMPARE LENGTHS
        BEQ     GOADD           ; EQUAL - O.K.
        JSR     PC,DECADJ       ; GO MAKE EQUAL
GOADD:  MOV     (R0),BK         ; BK HAS LENGTH (WORDS)
        ASL     (R0)            ; R0 POINTS TO LENGTH (BYTES)
        MOV     R0,R1           ; R1 POINTS TO OP 1 LSB + 1 WORD
        ADD     (R0),R0         ; R0 POINTS TO OP 2 LSB
        TST     SUBFLAG         ; ADD OR SUBTRACT?
        BEQ     ADLOOP
        BR      SUBBER
SBLOOP: SBC     -(R0)           ; CARRY
        BCC     $1
        SUB     -(R1),(R0)      ; IF HERE, MUST PASS ON CARRY
        SEC
        BR      SUB2
$1:     BVC     SUBBER          ; KEEP TRACK OF OVERFLOW
        SUB     -(R1),(R0)
```

```
         BR       SUB3
SUBBER:  SUB      -(R1),(R0)        ; PERFORM SUBTRACTION
         BVC      SUB2              ; SEPARATE LOOP-END FOR OVFL
SUB3:    SOB      BK,SBLOOP
         BCC      NWORD
         BR       ZWORD
SUB2:    SOB      BK,SBLOOP
         BR       NOXTRA            ; FINISHED SUBTRACTION W/O OVERFLOW
DAD1:    ADC      -(R0)             ; ADD CARRY BIT IN
         BCC      $1                ; IF ADDEND WAS -1 THEN RESULT IS 0 WITH
         MOV      -(R1),(R0)        ; CARRY, SO JUST MOVE 2ND ADDEND
         SOB      BK,DAD1           ; AND KEEP CARRY.
         BR       NOXTRA
$1:      BVC      ADLOOP            ; IF OVERFLOW THEN
         ADD      -(R1),(R0)        ; KEEP TRACK OF IT
         BR       OLOOP
ADLOOP:  ADD      -(R1),(R0)        ; MOVE R1 AND ADD
         BVS      OLOOP             ; SEPARATE LOOP-END FOR OVFL
         SOB      BK,DAD1           ; FOR EACH WORD PAIR
         BR       NOXTRA            ; EXTRA WORD NOT NEEDED
OLOOP:   SOB      BK,DAD1
         BCC      ZWORD             ; RESULT POSITIVE
NWORD:   MOV      #-1,-(R0)         ; RESULT NEGATIVE
         BR       PUTLEN
ZWORD:   CLR      -(R0)             ; PUT SIGN EXTENSION IN
PUTLEN:  ADD      #1,(SP)           ; INCREASE LENGTH BY ONE
NOXTRA:  MOV      (SP),-(R0)        ; PUT LENGTH IN RESULT
         MOV      R0,SP             ; ADJUST SP
         JMP      @(PC)+            ; RETURN
ASRET:   .WORD
SUBFLAG:.WORD                       ; ADD/SUBTRACT INDICATOR

DECADJ:  ; THIS ROUTINE MAKES 2 DECIMALS,
         ; (TOS) AND (TOS-1), OF EQUAL LENGTH.
         ; (SP)=UPPER LENGTH (WORDS)
         ; (R0)=LOWER LENGTH (WORDS)
         ; SAME CONDITIONS ON OUTPUT
         ; REGISTERS BK,R1 ARE DESTROYED
         MOV      (SP)+,DRET        ; SAVE RETURN ADDR
         MOV      BASE,BASSAV       ; SAVE REG
         MOV      @SP,R1
```

```
        SUB     @R0,R1          ; R1=LEN DIFF (WORDS) <> 0
        ASL     R1              ; CHANGE TO BYTES
        BLT     TOP             ; GO EXPAND TOP DECIMAL
        MOV     SP,BK           ; SAVE OLD TOS
        SUB     R1,SP           ; MAKE ROOM FOR EXPANSION
        MOV     SP,BASE         ; POINTS TO NEW TOS
SHIFT1: MOV     (BK)+,(BASE)+   ; SHIFT WORDS
        CMP     BK,R0           ; UNTIL ENTIRE TOP
        BLT     SHIFT1          ; OPERAND SHIFTED
        MOV     BASE,R0         ; POINT R0 AT BOTTOM LENGTH (FUTURE)
        MOV     (SP),(R0)       ; PUT IN BOTTOM LENGTH
        BR      FILL
TOP:    NEG     R1
        MOV     SP,BK           ; SAVE OLD TOS
        SUB     R1,SP           ; UPDATE SP
        MOV     (R0),(SP)       ; SET LENGTHS EQUAL
FILL:   TST     2(BK)           ; FILL WITH ZEROES OR ONES?
        SXT     BASE            ; SIGN EXTENSION
        ASR     R1              ; BACK TO WORDS
$1:     MOV     BASE,(BK)       ; MOVE FILLER
        TST     -(BK)           ; DECREMENT BY 2
        SOB     R1,$1           ; UNTIL FULL
        MOV     BASSAV,BASE     ; RESTORE REG
        JMP     @(PC)+          ; RETURN
DRET:   .WORD
BASSAV: .WORD

DCH:    ; CHECK DECIMAL LENGTH
        CMP     #10.,(SP)       ; CHECK LENGTH  (WORST CASE)
        BLT     DOVR            ; OVERFLOW IF TOO LONG
        JMP     @#BIGRTN

DOVR:   TRAP    INTOVR          ; OVERLFOW


DMP:    ; DECIMAL MULTIPLY
        JSR     PC,DMUL
        BR      DCH             ; CHECK FINAL LENGTH AND LEAVE

IPCSAV: .WORD
```

```
DMUL:   MOV     (SP)+,DMPRET    ; SAVE RETURN ADDR
        MOV     IPC,IPCSAV      ; R0-R5 USED
        CLR     NEG1            ; NEGATIVE REMEMBERER
        MOV     (SP)+,R0        ; POP OFF MULTIPLIER LENGTH
        TST     (SP)            ; CHECK FOR NEG SIGN
        BPL     $1
        MOV     R0,-(SP)        ; SETUP FOR CALL
        MOV     SP,R1
        JSR     PC,DODNG        ; ABS VALUE RETURNED
        INC     NEG1            ; REMEMBER
        MOV     (SP)+,R0        ; REPEAT OF ABOVE SETUP
$1:     TST     (SP)            ; GET RID OF LEADING ZEROES
        BNE     DGET
        TST     (SP)+           ; INCREMENT TO NEXT LEADING DIGIT
        SOB     R0,$1
        ; IF HERE THEN PRODUCT IS ZERO
        ; FILL MULTIPLICAND WITH ZEROES AND EXIT
ZPROD:  MOV     (SP),R0         ; PUT LENGTH IN R0
        MOV     SP,R1
        TST     (R1)+           ; INCREMENT R1 TO MSB
$1:     CLR     (R1)+
        SOB     R0,$1           ; CLEAR ALL WORDS
        JMP     DMPEND
DGET:   MOV     R0,R1           ; HERE GET RID OF MULTIPLICAND ZEROES
        ASL     R1              ; FIRST LOCATE LENGTH
        ADD     SP,R1           ; AND PUT IN R1
        MOV     (R1)+,R2        ; NOW R2 HAS LENGTH AND R1 POINTS TO MSB
        TST     (R1)            ; CHECK NEG.
        BPL     $1
        TST     -(R1)           ; DECREMENT TO POINT TO LENGTH
        JSR     PC,DODNG        ; FOR CALL TO ABS VALUE GETTER
        DEC     NEG1            ; IF BOTH OPS NEG THEN ZERO RESULTS
        MOV     (R1)+,R2        ; REPEAT OF ABOVE STMT
$1:     TST     (R1)            ; SAME LEADING ZERO PROCESSING
        BNE     DSETUP
        TST     (R1)+
        SOB     R2,$1
        ASL     R0              ; OOPS IT WAS ALL ZEROES
        ADD     R0,SP           ; ADJUST SP AND LEAVE
        JMP     DMPEND          ;
DSETUP: MOV     SP,R3           ; SP POINTS TO M'PLIER LSB..NEEDED LATER
```

```
        MOV     R2,R4           ; CONSTRUCT PRODUCT LENGTH NOW
        ADD     R0,R4           ; EQUALS SUM OF OPR. LENGTHS
        MOV     R4,LENSAV
$1:     CLR     -(SP)           ; CLEARING AREA FOR PRODUCT ON TOP OF STACK
        SOB     R4,$1
        MOV     R2,L2SAV        ; MULTIPLICAND LENGTH
        CLR     POSSAV          ; POSSAV=LEADING CURRENT POSITION IN PRODUCT
        MOV     R0,COUNT        ; MULTIPLIER LENGTH
        MOV     SP,R0           ; R0 WILL BE CURRENT PRODUCT WORD
        BR      DMULT
HILOOP: MOV     R4,COUNT        ; RE-SAVE COUNT OF M'PLIER WORDS
        MOV     POSSAV,R0       ; INCREMENT TO NEXT PRODUCT POSITION
        TST     (R0)+
        MOV     R0,POSSAV
        ADD     SP,R0           ; SET UP R0 TO POINT TO IT
        MOV     L2SAV,R2        ; REINIT LOOP COUNT
        SUB     R2,R1
        SUB     R2,R1           ; BACK TO BEGINNING OF M'CAND
        TST     (R3)+           ; KICK M'PLIER INDEX
DMULT:  MOV     (R3),R4         ; MULTIPLIER WORD
        MOV     (R1),R5         ; M'CAND WORD
        TST     R4              ; PERFORM TWO'S COMPLEMENT ADJUSTMENTS FIRST
        BPL     $1
        TST     R5
        BPL     $2
        ADD     R5,R4           ; BOTH NEG. - IGNORE OVERFLOW HERE
        BR      $3
$1:     CLR     ADJSAV          ; ENSURE CLEAR FOR NO ADJUSTMENT
        TST     R5
        BPL     DOOIT           ; BOTH POS.
        BR      $3              ; R5 NEG., ADD R4
$2:     MOV     R5,R4           ; R4 NEG., ADD R3
$3:     MOV     R4,ADJSAV       ; ADJUSTMENT FOR HI ORDER HALF OF PRODUCT
        MOV     (R3),R4
DOOIT:  MOV     (R1)+,R5
        JSR     PC,MLI          ; TIMES M'CAND WORD
        ADD     ADJSAV,R4       ; STICK IN ADJUSTMENT - IGNORE CARRY
        ADD     R4,(R0)+        ; HI-ORDER PROD
        BCC     $1
        DEC     R0              ; RE-ALIGN R0
        DEC     R0
```

```
        MOV     NNN,R3
$3:     CLR     -(SP)
        SOB     R3,$3           ; MAKE N WORDS OF ZEROS FOR B.
        MOV     AADD,R4
        MOV     MMM,R5
        TST     (R4)+
$4:     MOV     -(R4),-(SP)
        SOB     R5,$4           ; MOVE M WORDS FROM AADD TO TOP OF STACK

        .INCLUDE  DECOP.B.TEXT

        MOV     NNN,R5
        SUB     MMM,R5
        BEQ     $6
        BPL     $5
        JMP     BOMB
$5:     MOV     BSIGN,-(SP)
        SOB     R5,$5
$6:     MOV     CADD,R4
        MOV     AADD,R5
        CMP     (R4)+,(R5)+     ; ADJUST R4 AND R5
        MOV     NNN,R3
$7:     MOV     -(R4),-(R5)
        CLR     (R4)
        SOB     R3,$7           ; MOVE A AND CLEAR C.
        MOV     TWON,R3
        ASL     R3
        MOV     R3,LENG         ; SAVE LONG LENGTH IN BYTES
        TST     -(R3)
        MOV     AADD,AWS
        SUB     R3,AWS          ; SAVE ADDRESS OF SIGN WORD OF A.
        MOV     BADD,BWS
        SUB     R3,BWS          ; SAVE ADDRESS OF SIGN WORD OF B.
        MOV     AADD,R2
        MOV     TWON,R3
        JSR     PC,LASL
        MOV     AADD,R2
        MOV     TWON,R3
        JSR     PC,LASL         ; ADJUST A FOR PROPER ~VISION.
```

```
DDVB:   ;  DIVISION BEGINS HERE.
        CMP     ASIGN,BSIGN
        BEQ     $1
        JSR     PC,LADDAB       ; SIGN OF A IS NOT SAME AS SIGN OF B
        SEC                     ; SO A:=A+B AND ANSWER WILL BE NEGATIVE
        BR      $2
$1:     JSR     PC,LSUBAB       ; SIGN OF A SAME AS SIGN OF B SO A:=A-B
        CLC                     ; AND ANSWER WILL BE POSITIVE.
$2:     JSR     PC,LSLC         ; SHIFT CARRY LEFT INTO LSBIT OF C
        MOV     TWON,R5         ; ON A 16 BIT MACHINE THE NUMBER OF REPITITIONS
        ASL     R5              ; IS <NUMBER OF WORDS IN A> X
        ASL     R5              ;       <16 BITS PER WORD> - 2
        ASL     R5
        TST     -(R5)           ; R5 CONTAINS THE NUMBER OF REPITITIONS FOR
                                ; THE MAIN DIVISION LOOP.
MDL:    ;  MAIN DIVISION LOOP BEGINS HERE.
$1:     JSR     PC,LSRB         ; SHIFT TRIAL DIVISOR LEFT.
        TST     @AWS
        SXT     R0
SXTBRK: TST     @BWS
        SXT     R1
        CMP     R0,R1           ; IF EQUAL THEN PARTIAL REMAINDER AND TRIAL
        BEQ     $2              ; DIVISOR HAVE THE SAME SIGN.
        JSR     PC,LADDAB       ; SIGN OF PR AND TD DIFFERENT
                                ; SO PR:=PR+TD  (A:=A+B)
        CLC                     ; AND SHIFT A ZERO INTO LSBIT OF QUOTIENT
                                ; C:=(C$2)+0
        BR      $3
$2:     JSR     PC,LSUBAB       ; SIGN OF PR AND TD SAME SO PR:=PR-TD  (A:=A-B)
        SEC                     ; AND SHIFT A ONE INTO LSBIT OF Q.  C:=(C$2)+1
$3:     JSR     PC,LSLC         ; ACTUALLY DO THE SHIFT ON C.
        SOB     R5,MDL
EOMDL:  ;  END OF MAIN DIVISION LOOP
        SEC
        JSR     PC,LSLC         ; SHIFT A ONE INTO LSBIT OF C.
        TST     @AWS
        SXT     R5
        CMP     R5,ASIGN        ; IF SIGN OF A EQUALS SIGN OF REMAINDER THEN
        BEQ     $6              ; BRANCH TO $6 ELSE
        TST     ASIGN           ; IF A<0 THEN
        BNE     $3              ; ANSWER CORRECT AND DONE
```

```
        TST     BSIGN           ; OTHERWISE
        BEQ     $7              ; IF B>=0 THEN $7 ELSE
        BR      $8              ; IF B<0 THEN $8
$6:     TST     ASIGN
        BEQ     $3              ; IF A>=0 THEN DONE OTHERWISE
        TST     BSIGN
        BEQ     $7              ; IF B>=0 THEN $7 ELSE
        BR      $8              ; IF B<0 THEN $8.
$8:     MOV     NNN,R0          ; DO LONG ADD  C=C+1
        MOV     CADD,R1         ;  PREFORMED ONLY WHEN B>=0
        TST     (R1)+
        SEC
$2:     ADC     -(R1)
        BCC     $3              ; EARLY TERMINATION OF LONG ADDITION
        SOB     R0,$2
        BR      $3              ; END OF ADDITION    C=C+1
$7:     BIC     #1,@CADD        ; IF B<0 THEN CLEAR LSBIT OF C AND FINISHED.
$3:     MOV     AADD,SP         ; ADJUST STACK POINTER
        TST     (SP)+           ; AND LEAVE.
LEAVE:  JMP     @#BIGRTN          ; EXIT DECOPS

BOMB:   MOV     CADD,SP         ; THIS IS THE ERROR  CASE.  IT CURRENTLY
        CLR     (SP)            ; SETS THE ANSWER TO ZERO.  IN THE FUTURE IT
        MOV     #1,-(SP)        ; WILL GENERATE AN ERROR TRAP.
        BR      LEAVE

LSLC:   MOV     CADD,R2
        MOV     NNN,R3
        ROL     (R2)
        DEC     R3
        BEQ     EOSL
        BR      BSLC
LASL:           ; LONG ARITHMETIC SHIFT LEFT
        TST     (R2)+
BSLC:   ;
$1:     ROL     -(R2)
        SOB     R3,$1
EOSL:   RTS     PC              ;RETURN FROM ARITHMETIC SHIFT LEFT

LSRB:   MOV     BWS,R2
        MOV     TWON,R3
```

```
LASR:   ASR     (R2)+           ; LONG ARITHMETIC SHIFT RIGHT
        DEC     R3
        BEQ     EOSR
$1:     ROR     (R2)+
        SOB     R3,$1
EOSR:   RTS     PC              ; RETURN FROM ARITHMETIC SHIFT RIGHT

LADDAB: MOV     AADD,R0         ; A=A+B
        MOV     BADD,R1
        MOV     TWON,R2
LADD:   CMP     (R0)+,(R1)+     ; ADJUST ADDRESSES
$1:     ADD     -(R1),-(R0)
        MOV     R2,R3
        DEC     R3
        BEQ     $3
        MOV     R0,R4
$2:     BCC     $3
        ADC     -(R4)
        SOB     R3,$2
$3:     SOB     R2,$1
EOLADD: ;  END OF LONG ADD.
        RTS     PC

LSUBAB: MOV     AADD,R0         ; A=A-B
        MOV     BADD,R1
        MOV     TWON,R2
LSUB:   CMP     (R0)+,(R1)+     ; ADJUST ADDRESSES
$1:     SUB     -(R1),-(R0)
        MOV     R2,R3
        DEC     R3
        BEQ     $3
        MOV     R0,R4
$2:     BCC     $3
        SBC     -(R4)
        SOB     R3,$2
$3:     SOB     R2,$1
EOLSUB: ;  END OF LONG SUBTRACT.
        RTS     PC
        ; THIS IS THE  END OF DDV.
        ; FOLLOWING ARE STATIC STORAGE WORDS OF DDV.
```

```
MMM:    .WORD
NNN:    .WORD
TWON:   .WORD
LENG:   .WORD
AWS:    .WORD
BWS:    .WORD
ASIGN:  .WORD
BSIGN:  .WORD
AADD:   .WORD
BADD:   .WORD
CADD:   .WORD
DVIPC:  .WORD


DECCMP: ; COMPARE DECIMALS
        MOV     (SP)+,R0        ; GET COMPARISON TYPE INDEX
        ASL     R0
        MOV     SBROPS(R0),$5   ; PUT IN SIGNED CMP OPR
        MOV     UBROPS(R0),$2   ; PUT IN UNSIGNED OPR
        MOV     (SP),R0         ; PROCESSING TO POINT R0 AT LEFT OPR LENGTH
        ASL     R0
        ADD     SP,R0
        TST     (R0)+           ; R0 NOW POINTS TO LENGTH
        CMP     @R0,@SP         ; COMPARE LENGTHS
        BEQ     $8
        JSR     PC,DECADJ       ; MAKE LENGTHS EQUAL
$8:     MOV     (R0),BASE       ; BASE WILL HOLD ADDR OF RESULT
        ASL     BASE
        ADD     R0,BASE         ; BASE NOW POINTS TO RESULT
        TST     (R0)+           ; R0 POINTS TO LEFT OPR MSB
        MOV     (SP)+,BK        ; SP POINTS TO RIGHT OPR MSB; BK HAS LENGTH
        CMP     (R0)+,(SP)+     ; COMPARE SIGN WORDS
        BNE     $5
        BR      $7
$1:     CMP     (R0)+,(SP)+     ; COMPARE UNSIGNED WORDS
        BNE     $2              ; ANY NEQ STOPS LOOP
$7:     SOB     BK,$1
$2:     NOP                     ; UNSIGNED COMPARE OP GOES HERE
        BR      $4
        BR      $6
$5:     NOP                     ; SIGNED CMP OP HERE
```

```
        BR      $4
$6:     MOV     #1,(BASE)
$3:     MOV     BASE,SP
        JMP     @#BIGRTN        ; EXIT DECOPS

$4:     CLR     (BASE)
        BR      $3

DSTR:   ; DECIMAL TO STRING CONVERT
                                ; REGISTER ASSIGNMENTS:
                                ; IPC = POWER OF TEN INDEX
                                ; BASE = POINTER TO STRING
                                ; MP = POINTER TO LAST CHAR IN STRING
                                ; BK = DECIMAL LENGTH
        MOV     (SP)+,SLENG
        MOV     (SP)+,BASE
        MOV     BASE,MP
        INC     MP
        MOVB    #1,(BASE)
        MOVB    #"0",(MP)       ; INITIALIZE
        CLR     SIGNIF          ; SIGNIFICANCE FLAG
        CLR     ZCOUNT          ; SHIFT COUNTER
        MOV     SP,R1           ; FOR DODNG AND LATER PROCESSING
        TST     2(SP)           ; NEGATIVE?
        BGE     $1
        JSR     PC,DODNG        ; MAKE POSITIVE
        MOVB    #"-",(MP)       ; INSERT MINUS SIGN
        INCB    (BASE)
        INC     MP
        MOVB    #"0",(MP)
$1:     MOV     (SP),BK
        TST     (R1)+           ; INCREMENT TO SIGN WORD
$2:     TST     (R1)+           ; GET RID OF LEADING ZEROES
        BNE     $3              ; -(R1) WAS NONZERO
        SOB     BK,$2           ; BK KEEPS TRACK OF LENGTH
        MOV     R1,SP           ; IF HERE, NUMBER WAS ZERO
        BR      STREND          ; SO POP DECIMAL AND LEAVE
$3:     TST     -(R1)           ; CHECK SIGN BIT - SHOULD BE POSITIVE
        BGE     $4
        TST     -(R1)           ; RESTORE A WORD OF ZEROES
        INC     BK
```

```
$4:     MOV     BK,-(R1)        ; PUT ON LENGTH TO MAKE
        MOV     R1,SP           ; A COMPLETE (SHORTER) DECIMAL
        ; NOW CHOOSE APPROPRIATE POWER OF TEN
        CMP     BK,#2           ; DECIMAL LEN <= 2 ?
        BGT     C4
        MOV     #POT2,IPC       ; YES, POINT IPC TO RIGHT P.O.T.
        BR      CEND
C4:     CMP     BK,#4           ; DECIMAL LEN <= 4 ?
        BGT     C8
        MOV     #POT4,IPC
        BR      CEND
C8:     MOV     #POT8,IPC
CEND:   ; HERE COMPARE INPUT TO  POT
        CMP     (SP),(IPC)      ; COMPARE LENGTHS
        BGT     $1              ; DECIMAL LONGER (GREATER)
        BLT     DLESS           ; DECIMAL SHORTER (SMALLER)
        CMP     2(SP),2(IPC)    ; COMPARE MSB WORDS
        BLO     DLESS           ; DECIMAL SMALLER
        ; ELSE ASSUME DECIMAL GREATER OR EQUAL FOR NOW
$1:     JSR     PC,CLOAD        ; PUT POT ON STACK
        MOV     SP,SUBFLAG      ; SET NONZERO FLAG
        JSR     PC,ADDSUB       ; AND SUBTRACT (NOTE NO REG SAVE)
        ; TEST FOR ZERO OR LESS
        TST     2(SP)
        BLT     RESTOR          ; OOPS DECIMAL WAS SMALLER
        INCB    (MP)            ; ALSO INCREMENT STRING HERE
        INC     SIGNIF          ; AND SET FLAG
        MOV     (SP),R1         ; LENGTH
        TST     (SP)+           ; INCREMENT
$2:     TST     (SP)            ; LOOP TO TEST FOR ZERO
        BLT     $3              ; RESULT AND ALSO
        BNE     $4              ; SHORTEN DECIMAL
        TST     (SP)+           ; BY REMOVING LEADING ZEROES
        SOB     R1,$2
        BR      TRAIL           ; DIFFERENCE ZERO - CONVERSION COMPLETE
$3:     CLR     -(SP)           ; HERE, RETAIN EXTRA LEADING
        INC     R1              ; ZEROES FOR POSITIVE SIGN
$4:     MOV     R1,-(SP)        ; LENGTH
        BR      CEND            ; DO IT AGAIN
RESTOR: JSR     PC,CLOAD
        CLR     SUBFLAG         ; INDICATES ADD
```

```
        JSR     PC,ADDSUB       ; FALLS INTO DLESS
DLESS:  MOV     MP,CHSAV        ; SAVE REGS EXCEPT IPC
        MOV     BASE,BASESV
        MOV     #10.,-(SP)      ; PUSH D 10 ON STACK
        MOV     #1,-(SP)
        JSR     PC,DMUL         ; AND MULTIPLY
        MOV     CHSAV,MP        ; RESTORE
        MOV     BASESV,BASE
        INC     ZCOUNT          ; COUNT THE SHIFT
        TST     SIGNIF          ; DON'T PUT OUT A NON-SIGNIF. ZERO
        BEQ     CEND
        INCB    (BASE)
        INC     MP
        MOVB    #"0",(MP)
        BR      CEND            ; REPEAT PROCESS FOR NEXT DIGIT
TRAIL:  ; HERE, ADD TRAILING BLANKS
        MOV     -(IPC),R0       ; NO. OF ZEROES IN POT
        SUB     ZCOUNT,R0       ; THESE DIGITS ACCOUNTED FOR
        BEQ     STREND          ; IF NO TRAILING ZEROES
$1:     INCB    (BASE)
        INC     MP
        MOVB    #"0",(MP)
        SOB     R0,$1
STREND: CMPB    @BASE,SLENG     ; CHECK STRING LENGTH
        BLOS    $1
        TRAP    S2LONG
$1:     JMP     @#BIGRTN        ; EXIT DECOPS


CLOAD:  ; LOAD A POWER OF TEN
        MOV     (SP)+,$3        ; SAVE RETURN ADDR
        MOV     (IPC),R0        ; LENGTH
        INC     R0              ; INCLUDE LENGTH IN TRANSFER
        MOV     IPC,R1
        ADD     R0,R1
        ADD     R0,R1           ; NOW R1 POINTS TO LSB +  2
$1:     MOV     -(R1),-(SP)
        SOB     R0,$1
        TST     2(SP)
        BGE     $2              ; IF SIGN BIT USED
        MOV     (SP),-(SP)      ; THEN MAKE POSITIVE
```

```
        CLR     2(SP)
        INC     (SP)
$2:     JMP     @(PC)+          ; RETURN
$3:
CHSAV:  .WORD                   ; DOUBLE DUTY
BASESV: .WORD
SLENG:  .WORD
SIGNIF: .WORD
ZCOUNT: .WORD
POT2    .EQU    * + 2           ; STORAGE FOR 10**9 (MAX 2-WORD PWR OF TEN)
        .WORD   9.              ; NUMBER OF 0'S IN 10**9
        .WORD   2               ; LENGTH
        .WORD   035632          ; OCTAL REPRESENTATION (HI-ORDER FIRST)
        .WORD   145000
POT4    .EQU    * + 2           ; SAME FOR  10**19 (4 WORDS)
        .WORD   19.
        .WORD   4
        .WORD   105307          ; NOT TWOS COMPLEMENT!!
        .WORD   021404
        .WORD   104750
        .WORD   000000
POT8    .EQU  * + 2                     ; SAME FOR 10**36 (8 WORDS)
; NOTE ANY NUMBERS GREATER THAN OR EQUAL TO 10**37 BUT 8 WORDS LONG
; ARE MISREPRESENTED
        .WORD   38.
        .WORD   8.
        .WORD   045473
        .WORD   046250
        .WORD   055206
        .WORD   142172
        .WORD   004612
        .WORD   021100
        .WORD   000000
        .WORD   000000

DCV:    ; CONVERT INTEGER TO DECIMAL AT NEXT-TO-TOS
        ; TOS MUST BE A DECIMAL
        MOV     (SP),R0         ; LENGTH IN R0
        MOV     SP,BK           ; DESTINATION POINTER FOR MOVE
        MOV     BK,R1           ; R1 IS
        TST     (R1)+           ; SOURCE POINTER
```

```
        MOV     (SP),-(SP)      ; MOVE LENGTH
$1:     MOV     (R1)+,(BK)+     ; MOVE DECIMAL
        SOB     R0,$1
        MOV     #1,(BK)         ; LENGTH WORD FOR INTEGER
        JMP     @#BIGRTN        ; EXIT DECOPS


SBROPS  .EQU    * - 16.
        BLT     * + 4
        BLE     * + 4
        BGE     * + 4
        BGT     * + 4
        BNE     * + 4
        BEQ     * + 4

UBROPS  .EQU    * - 16.
        BLO     * + 4
        BLOS    * + 4
        BHIS    * + 4
        BHI     * + 4
        BNE     * + 4
        BEQ     * + 4


        .END
```

```
 1    1     1:0     1 (*$L PRINTER:*)
 2    1     1:0     1 PROGRAM CODESTAT;
 3    1     1:0     3
 4    1     1:0     3 [============================================================]
 5    1     1:0     3 [                                                            ]
 6    1     1:0     3 [        UCSD     P-CODE     DISASSEMBLER   II.0 [A.5]      ]
 7    1     1:0     3 [                                                            ]
 8    1     1:0     3 [        RELEASE LEVEL:     II.0    SEPT, 1978              ]
 9    1     1:0     3 [                                                            ]
10    1     1:0     3 [        WRITTEN BY     WILLIAM  P.  FRANKS                  ]
11    1     1:0     3 [                                                            ]
12    1     1:0     3 [        INSTITUTE FOR INFORMATION SYSTEMS                   ]
13    1     1:0     3 [        UC  SAN DIEGO,  LA JOLLA,  CA                       ]
14    1     1:0     3 [                                                            ]
15    1     1:0     3 [        KENNETH  L.  BOWLES,  DIRECTOR                      ]
16    1     1:0     3 [                                                            ]
17    1     1:0     3 [        COPYRIGHT  (C)  1978,  REGENTS OF THE               ]
18    1     1:0     3 [        UNIVERSITY OF CALIFORNIA,  SAN DIEGO                ]
19    1     1:0     3 [                                                            ]
20    1     1:0     3 [============================================================]
21    1     1:0     3
22    1     1:0     3 CONST    VERSION='II.0 [A.5]';
23    1     1:0     3          MAXPROCNUM=150;
24    1     1:0     3
25    1     1:0     3 TYPE     NMENONIC=PACKED ARRAY[0..7] OF CHAR;
26    1     1:0     3          BYTETYPE=ARRAY[0..7] OF INTEGER;
27    1     1:0     3          WORDTYPE=ARRAY[0..15] OF INTEGER;
28    1     1:0     3          BYTE=0..255;
29    1     1:0     3          OPTYPE=(SHORT,ONE,OPT,TWO,LOPT,WORDS,CHRS,BLK,CMPRSS,CMPRSS2,WORD);
30    1     1:0     3          OPREC=RECORD CASE OPTYPE OF
31    1     1:0     3                  SHORT:(TOTAL0:INTEGER);
32    1     1:0     3            ONE,CHRS,BLK:(TOTAL1:INTEGER;
33    1     1:0     3                          BYTEONE1:BYTETYPE);
34    1     1:0     3                   TWO:(TOTAL2:INTEGER;
35    1     1:0     3                          BYTEONE2:BYTETYPE;
36    1     1:0     3                          BYTETWO2:BYTETYPE;
37    1     1:0     3                          FLAVOR2:ARRAY[2..29] OF INTEGER);
38    1     1:0     3               WORD,OPT:(TOTAL3:INTEGER;
39    1     1:0     3                          PARMONE3:WORDTYPE);
40    1     1:0     3                  LOPT:(TOTAL4:INTEGER;
41    1     1:0     3                          BYTEO   :BYTETYPE;
```

```
42   1    1:0    3                    PARMTWO4:WORDTYPE);
43   1    1:0    3              WORDS:(TOTAL5:INTEGER;
44   1    1:0    3                    PARMONE5:WORDTYPE;
45   1    1:0    3                    PARMTWO5:WORDTYPE;
46   1    1:0    3                    PARMTHREE5:WORDTYPE);
47   1    1:0    3             CMPRSS:(TOTAL6:INTEGER;
48   1    1:0    3                    FLAVOR6:ARRAY[0..40] OF INTEGER);
49   1    1:0    3           CMPRSS2:(TOTAL7:INTEGER;
50   1    1:0    3                    FLAVOR7:ARRAY[1..6] OF INTEGER)
51   1    1:0    3       END;
52   1    1:0    3     OPPTR=^OPREC;
53   1    1:0    3     OPFACTS=RECORD
54   1    1:0    3       NAMES:ARRAY[52..255] OF NMENONIC;
55   1    1:0    3       RECTYPES:ARRAY[0..255] OF OPTYPE
56   1    1:0    3     END;
57   1    1:0    3     JUMPREC=RECORD
58   1    1:0    3       POS,NEG:WORDTYPE
59   1    1:0    3     END;
60   1    1:0    3     PRCLARRY=ARRAY[0..MAXPROCNUM] OF INTEGER;
61   1    1:0    3     DSPTR=^DSARRY;
62   1    1:0    3     DSARRY=ARRAY[0..1] OF INTEGER;
63   1    1:0    3     HEXTYPE=PACKED RECORD CASE INTEGER OF
64   1    1:0    3         0:(DUM2,DUM1,HI,LO:0..15);
65   1    1:0    3         1:(HIBYTE,LOWBYTE:0..255);
66   1    1:0    3         2:(WORD:INTEGER)
67   1    1:0    3       END;
68   1    1:0    3
69   1    1:0    3  VAR  DISPLAY:BOOLEAN;
70   1    1:0    4     CH,CR:CHAR;
71   1    1:0    6     PCTMAX,MAXOP,INUM,BYTESIZE,BYTEPOS,OP,BUFSTART,PROCNUM,SEGNUM:INTEGER;
72   1    1:0   15     BITE:BYTE;
73   1    1:0   16     OSSTART:DSPTR;
74   1    1:0   17     SWAP,CONTROL,CONSOLE,DONEPROC,LEXCHECK,DATAWATCH,
75   1    1:0   17     LEXLOOK   :BOOLEAN;
76   1    1:0   24     HEXCOUNT,MAXPROC,SEGSTBLK,BUFSTBLK,OPTOTAL,
77   1    1:0   24     SEGSIZE,OFFSET,BACKJUMP,SLDC,
78   1    1:0   24     SLDL,SLDO,SIND,PROCSTART,DATASEG,DATAPROC,
79   1    1:0   24     DATASEGSIZE,LEXLEVEL,DATAREF,DTSGSZ,JUMPTOTAL   :INTEGER;
80   1    1:0   44     HEX:HEXTYPE;
81   1    1:0   45     RNUM:REAL;
82   1    1:0   47     OPCODE:ARRAY[0..255] OF OPPTR;
```

```
 83    1     1:0    303          LISTFILE:INTERACTIVE;
 84    1     1:0    604          HEXCHAR,CODE   :PACKED ARRAY[0..15] OF CHAR;
 85    1     1:0    620          INPUTFILE:FILE;
 86    1     1:0    660          JUMPSTATS:JUMPREC;
 87    1     1:0    692          SEGLEX:ARRAY[0..15] OF INTEGER;
 88    1     1:0    708          SEGDIREC:PACKED ARRAY[0..511] OF BYTE;
 89    1     1:0    964          NAMES:ARRAY[52..255] OF NMENONIC;
 90    1     1:0   1780          RECTYPES:PACKED ARRAY[0..255] OF OPTYPE;
 91    1     1:0   1844          PROCS:ARRAY [0..MAXPROCNUM] OF INTEGER;
 92    1     1:0   1995          PROCCALL:ARRAY[0..15] OF ^PRCLARRY;
 93    1     1:0   2011          JUMPS,PROCLEX:ARRAY[0..99] OF INTEGER;
 94    1     1:0   2211          LASTFILENAME:STRING;
 95    1     1:0   2252          BUFFER:PACKED ARRAY[0..2559] OF BYTE;
 96    1     1:0   3532
 97   10     1:0      1 SEGMENT PROCEDURE INIT;
 98   10     1:0      1 TYPE  RECTIFIER=RECORD CASE BOOLEAN OF
 99   10     1:0      1           TRUE:(INT:INTEGER);
100   10     1:0      1           FALSE:(RECTYPE:OPTYPE)
101   10     1:0      1         END;
102   10     1:0      1 VAR     I:INTEGER;
103   10     1:0      2         TESTTYPE:RECTIFIER;
104   10     1:0      3         FILENAME:STRING;
105   10     1:0     44         OPFILE:FILE OF OPFACTS;
106   10     1:0   1416
107   10     2:0      1 PROCEDURE NEWOP(FLAVOR:OPTYPE);
108   10     2:0      0 BEGIN
109   10     2:1      0   CASE FLAVOR OF
110   10     2:1      3          SHORT:NEW(OPCODE[I],SHORT);
111   10     2:1     20            ONE:NEW(OPCODE[I],ONE);
112   10     2:1     37            BLK:NEW(OPCODE[I],BLK);
113   10     2:1     54           CHRS:NEW(OPCODE[I],CHRS);
114   10     2:1     71            OPT:NEW(OPCODE[I],OPT);
115   10     2:1     88            TWO:NEW(OPCODE[I],TWO);
116   10     2:1    105           LOPT:NEW(OPCODE[I],LOPT);
117   10     2:1    122          WORDS:NEW(OPCODE[I],WORDS);
118   10     2:1    139         CMPRSS:NEW(OPCODE[I],CMPRSS);
119   10     2:1    156        CMPRSS2:NEW(OPCODE[I],CMPRSS2);
120   10     2:1    173           WORD:NEW(OPCODE[I],WORD)
121   10     2:1    188     END;
122   10     2:1    220   WITH OPCODE[I]^ DO
123   10     2:2    235     CASE FLAVOR OF
```

```
124  10    2:2   238                SHORT:TOTAL0:=0;
125  10    2:2   243        CHRS,BLK,ONE:BEGIN
126  10    2:4   243                       TOTAL1:=0;
127  10    2:4   246                       FILLCHAR(BYTEONE1,16,0);
128  10    2:3   254                     END;
129  10    2:2   256                 TWO:BEGIN
130  10    2:4   256                       TOTAL2:=0;
131  10    2:4   259                       FILLCHAR(BYTEONE2,16,0);
132  10    2:4   267                       FILLCHAR(BYTETWO2,16,0);
133  10    2:4   275                       FILLCHAR(FLAVOR2,56,0);
134  10    2:3   283                     END;
135  10    2:2   285             WORD,OPT:BEGIN
136  10    2:4   285                       TOTAL3:=0;
137  10    2:4   288                       FILLCHAR(PARMONE3,32,0);
138  10    2:3   296                     END;
139  10    2:2   298                LOPT:BEGIN
140  10    2:4   298                       TOTAL4:=0;
141  10    2:4   301                       FILLCHAR(BYTEONE4,16,0);
142  10    2:4   309                       FILLCHAR(PARMTWO4,32,0);
143  10    2:3   317                     END;
144  10    2:2   319               WORDS:BEGIN
145  10    2:4   319                       TOTAL5:=0;
146  10    2:4   322                       FILLCHAR(PARMONE5,32,0);
147  10    2:4   330                       FILLCHAR(PARMTWO5,32,0);
148  10    2:4   338                       FILLCHAR(PARMTHREE5,32,0);
149  10    2:3   346                     END;
150  10    2:2   348              CMPRSS:BEGIN
151  10    2:4   348                       TOTAL6:=0;
152  10    2:4   351                       FILLCHAR(FLAVOR6,82,0);
153  10    2:3   359                     END;
154  10    2:2   361             CMPRSS2:BEGIN
155  10    2:4   361                       TOTAL7:=0;
156  10    2:4   364                       FILLCHAR(FLAVOR7,12,0);
157  10    2:3   372                     END
158  10    2:2   372      END;
159  10    2:0   404 END;
160  10    2:0   424
161  10    1:0     0 BEGIN(* INIT *)
162  10    1:1     0   CR:=CHR(13);
163  10    1:1    14   (*$I-*)
164  10    1:1    14   RESET(OPFILE,'OPCODES.II.0[illegible];
```

```
165  10    1:1    36   IF IORESULT<>0 THEN
166  10    1:2    42     BEGIN
167  10    1:3    42       WRITELN('*OPCODES.I5 NOT ON SYSTEM DISK');
168  10    1:3    88       EXIT(CODESTAT);
169  10    1:2    92     END;
170  10    1:2    92   (*$I+*)
171  10    1:1    92   NAMES:=OPFILE^.NAMES;
172  10    1:1   100   FOR I:=0 TO 255 DO
173  10    1:2   116     BEGIN
174  10    1:3   116       NEWOP(OPFILE^.RECTYPES[I]);
175  10    1:3   132       IF ORD(OPFILE^.RECTYPES[I])>255 THEN
176  10    1:4   152         TESTTYPE.INT:=ORD(OPFILE^.RECTYPES[I]) MOD 256
177  10    1:3   166       ELSE
178  10    1:4   174         TESTTYPE.RECTYPE:=OPFILE^.RECTYPES[I];
179  10    1:3   190       RECTYPES[I]:=TESTTYPE.RECTYPE;
180  10    1:2   204     END;
181  10    1:1   211   CLOSE(OPFILE);
182  10    1:1   219   PAGE(OUTPUT);
183  10    1:1   229   GOTOXY(22,10);
184  10    1:1   234   WRITELN('UCSD   P-CODE   DISASSEMBLER ',VERSION);
185  10    1:1   305   GOTOXY(0,0);
186  10    1:1   310   WRITE('INPUT CODE FILE: ');
187  10    1:1   339   READLN(FILENAME);
188  10    1:1   358   (*$I-*)
189  10    1:1   358   OPENOLD(INPUTFILE,CONCAT(FILENAME,'.CODE'));
190  10    1:1   397   (*$I+*)
191  10    1:1   397   IF IORESULT <> 0 THEN
192  10    1:2   403      OPENOLD(INPUTFILE,FILENAME);
193  10    1:1   415   IF BLOCKREAD(INPUTFILE,SEGDIREC,1)=1  THEN ;
194  10    1:1   439   FOR SEGNUM:=0 TO 15 DO
195  10    1:2   453     IF SEGDIREC[SEGNUM*4] + SEGDIREC[SEGNUM*4 + 1]<>0 THEN
196  10    1:3   484       BEGIN
197  10    1:4   484         NEW(PROCCALL[SEGNUM]);
198  10    1:4   498         FILLCHAR(PROCCALL[SEGNUM]^,SIZEOF(PRCLARRY),0);
199  10    1:3   515       END
200  10    1:2   515     ELSE PROCCALL[SEGNUM]:=NIL;
201  10    1:1   535   PAGE(OUTPUT);
202  10    1:1   545   GOTOXY(0,10);
203  10    1:1   550   WRITELN(' ':10,'IS THIS CODE FILE DESIGNED FOR A MACHINE');
204  10    1:1   620   WRITE(' ':7,'WHERE BYTE ZERO IS THE MOST SIGNIFICANT BYTE <LSI-11 NO>?');
205  10    1:1   699   READ(KEYBOARD,CH);
```

```
206  10    1:1   709   SWAP:=(CH='Y') OR (CH='Y');
207  10    1:1   718   PAGE(OUTPUT);
208  10    1:1   728   GOTOXY(0,10);
209  10    1:1   733   WRITE('DIS-ASSEMBLY OUTPUT FILE (<CR> FOR NONE): ');
210  10    1:1   787   READLN(FILENAME);
211  10    1:1   806   LASTFILENAME:=FILENAME;
212  10    1:1   813   DISPLAY:=(FILENAME<>'');
213  10    1:1   822   CONSOLE:=(FILENAME='CONSOLE:') OR (FILENAME='#1:');
214  10    1:1   850   IF DISPLAY THEN REWRITE(LISTFILE,FILENAME);
215  10    1:1   865   SEGNUM:=0;
216  10    1:1   868   OPTOTAL:=0;
217  10    1:1   871   SLDC:=0;
218  10    1:1   874   SLDL:=0;
219  10    1:1   877   SLDO:=0;
220  10    1:1   880   SIND:=0;
221  10    1:1   883   JUMPTOTAL:=0;
222  10    1:1   886   HEXCOUNT:=0;
223  10    1:1   889   CODE:='                    ';
224  10    1:1   913   HEXCHAR:='0123456789ABCDEF';
225  10    1:1   937   FILLCHAR(JUMPSTATS.POS,32,0);
226  10    1:1   945   FILLCHAR(JUMPSTATS.NEG,32,0);
227  10    1:1   953   LEXLOOK:=FALSE;
228  10    1:0   956 END;
229  10    1:0   978
230   1    2:D     1 PROCEDURE PROMPT; FORWARD;
231   1    2:D     1
232  11    1:D     1 SEGMENT PROCEDURE DISASSEMBLE;
233  11    1:D     1
234  11    2:D     3 FUNCTION BUFRESET(BYTEPOS,OFFSET,DIRECTION:INTEGER):INTEGER;
235  11    2:D     6 VAR   NEWBYTE:INTEGER;
236  11    2:0     0 BEGIN
237  11    2:1     0   NEWBYTE:=BYTEPOS + OFFSET;
238  11    2:1     5   REPEAT
239  11    2:2     5     BUFSTBLK:=BUFSTBLK + DIRECTION;
240  11    2:2    11     BUFSTART:=(BUFSTBLK - SEGSTBLK)*512;
241  11    2:1    22   UNTIL (NEWBYTE - BUFSTART>=0) AND (NEWBYTE - BUFSTART<2557);
242  11    2:1    37   IF BLOCKREAD(INPUTFILE,BUFFER,5,BUFSTBLK)=1 THEN;
243  11    2:1    59   BUFRESET:=NEWBYTE - BUFSTART;
244  11    2:0    64 END;
245  11    2:0    78
246  11    3:D     3 FUNCTION LASTBYTE:BYTE;
```

```
247  11    3:D     3 VAR   CHANGE:INTEGER;
248  11    3:0     0 BEGIN
249  11    3:1     0   IF BYTEPOS<1 THEN
250  11    3:2     5     BEGIN
251  11    3:3     5       BYTEPOS:=BUFRESET(BUFSTART + BYTEPOS,-1,-1);
252  11    3:3    18       OFFSET:=OFFSET - 1;
253  11    3:2    24     END
254  11    3:1    24   ELSE
255  11    3:2    26     BEGIN
256  11    3:3    26       BYTEPOS:=BYTEPOS - 1;
257  11    3:3    31       OFFSET:=OFFSET - 1;
258  11    3:2    37     END;
259  11    3:1    37   LASTBYTE:=BUFFER[BYTEPOS];
260  11    3:0    54 END;
261  11    3:0    66
262  11    4:D     3 FUNCTION GETBYTE:BYTE;
263  11    4:D     3 VAR  HEX:HEXTYPE;
264  11    4:0     0 BEGIN
265  11    4:1     0   IF BYTEPOS>2559 THEN
266  11    4:2     7     BYTEPOS:=BUFRESET(BUFSTART + BYTEPOS,0,5);
267  11    4:1    18   GETBYTE:=BUFFER[BYTEPOS];
268  11    4:1    35   IF HEXCOUNT<15 THEN
269  11    4:2    41     BEGIN
270  11    4:3    41       HEX.LOWBYTE:=BUFFER[BYTEPOS];
271  11    4:3    61       CODE[HEXCOUNT]:=HEXCHAR[HEX.HI];
272  11    4:3    82       CODE[HEXCOUNT + 1]:=HEXCHAR[HEX.LO];
273  11    4:3   105       HEXCOUNT:=HEXCOUNT + 2;
274  11    4:2   111     END;
275  11    4:1   111   BYTEPOS:=BYTEPOS + 1;
276  11    4:0   116 END;
277  11    4:0   128
278  11    5:D     3 FUNCTION GETBIG:INTEGER;
279  11    5:D     3 VAR  BIG:HEXTYPE;
280  11    5:D     4      FIRSTBYTE:BYTE;
281  11    5:0     0 BEGIN
282  11    5:1     0   FIRSTBYTE:=GETBYTE;
283  11    5:1    11   IF FIRSTBYTE>127 THEN
284  11    5:2    16     BEGIN
285  11    5:3    16       BIG.LOWBYTE:=GETBYTE;
286  11    5:3    30       BIG.HIBYTE:=FIRSTBYTE - 128;
287  11    5:3    45       GETBIG:=BIG.WORD;
```

```
288  11     5:2     48     END
289  11     5:1     48   ELSE GETBIG:=FIRSTBYTE;
290  11     5:0     53 END;
291  11     5:0     66
292  11     6:D      3 FUNCTION GETWORD:INTEGER;
293  11     6:D      3 VAR  WERD:HEXTYPE;
294  11     6:0      0 BEGIN
295  11     6:1      0   IF SWAP THEN
296  11     6:2      4     BEGIN
297  11     6:3      4       WERD.HIBYTE:=GETBYTE;
298  11     6:3     18       WERD.LOWBYTE:=GETBYTE;
299  11     6:2     32     END
300  11     6:1     32   ELSE
301  11     6:2     34     BEGIN
302  11     6:3     34       WERD.LOWBYTE:=GETBYTE;
303  11     6:3     48       WERD.HIBYTE:=GETBYTE;
304  11     6:2     62     END;
305  11     6:1     62   GETWORD:=WERD.WORD;
306  11     6:0     65 END;
307  11     6:0     78
308  11     7:D      3 FUNCTION MOSTSIGBIT(OPERAND:INTEGER):INTEGER;
309  11     7:D      4 VAR  BYTESIZE:INTEGER;
310  11     7:0      0 BEGIN
311  11     7:1      0   IF OPERAND<0 THEN
312  11     7:2      5     MOSTSIGBIT:=15
313  11     7:1      5   ELSE
314  11     7:2     10     BEGIN
315  11     7:3     10       BYTESIZE:=-1;
316  11     7:3     14       REPEAT
317  11     7:4     14         BYTESIZE:=BYTESIZE + 1;
318  11     7:4     19         OPERAND:=OPERAND DIV 2;
319  11     7:3     24       UNTIL OPERAND=0;
320  11     7:3     29       MOSTSIGBIT:=BYTESIZE;
321  11     7:2     32     END;
322  11     7:0     32 END;
323  11     7:0     46
324  11     8:D      1 PROCEDURE ACTACCESS(FINALEX,OFFSET:INTEGER); FORWARD;
325  11     8:D      3
326  11     9:D      1 PROCEDURE SHORTOP;
327  11     9:D      1 [SLDC ABI  ABR  ADI  ADR  LAND DIF  DVI  DVR  CHK  FLO  FLT  INN  INT
328  11     9:D      1  LOR  MODI MPI  MPR  NGI  NOR  LNOT SRS  SBI  SBR  SGS  SQI  SQR  STO
```

```
329  11     9:D      1  IXS  UNI  S2P  LDCN LDP  STP  LDB  STB  EQUI GEQI GTRI LEQI LESI NEQI
330  11     9:D      1  S1P  IXB  BYT  XIT  SLDL SLDO SIND]
331  11     9:D      1
332  11     9:0      0 BEGIN
333  11     9:1      0   OPCODE[BITE]^.TOTAL0:=OPCODE[BITE]^.TOTAL0 + 1;
334  11     9:1     26   IF BITE=214 THEN DONEPROC:=TRUE;
335  11     9:1     36   IF BITE<128 THEN
336  11     9:2     43     BEGIN
337  11     9:3     43       SLOC:=SLOC + 1;
338  11     9:3     49       IF DISPLAY THEN WRITELN(LISTFILE,NAMES[127],BITE:6,' ':18,CODE);
339  11     9:2    116     END
340  11     9:1    116   ELSE
341  11     9:2    118     BEGIN
342  11     9:3    118       IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
343  11     9:3    144       IF BITE>215 THEN
344  11     9:4    151         IF BITE<232 THEN
345  11     9:5    158           BEGIN
346  11     9:6    158             SLDL:=SLDL + 1;
347  11     9:6    164             IF DATAWATCH THEN ACTACCESS(LEXLEVEL,BITE - 215);
348  11     9:6    177             IF DISPLAY THEN WRITELN(LISTFILE,BITE-215:6,' ':18,CODE);
349  11     9:5    225           END
350  11     9:4    225         ELSE IF BITE<248 THEN
351  11     9:6    234           BEGIN
352  11     9:7    234             SLDO:=SLDO + 1;
353  11     9:7    240             IF DATAWATCH THEN ACTACCESS(0,BITE - 231);
354  11     9:7    252             IF DISPLAY THEN WRITELN(LISTFILE,BITE-231:6,' ':18,CODE);
355  11     9:6    300           END
356  11     9:5    300         ELSE
357  11     9:6    302           BEGIN
358  11     9:7    302             SIND:=SIND + 1;
359  11     9:7    308             IF DISPLAY THEN WRITELN(LISTFILE,BITE-248:6,' ':18,CODE);
360  11     9:6    356           END
361  11     9:3    356       ELSE
362  11     9:4    358         IF DISPLAY THEN WRITELN(LISTFILE,' ':24,CODE);
363  11     9:2    392     END;
364  11     9:1    392   IF DONEPROC THEN
365  11     9:2    396     IF DISPLAY THEN WRITELN(LISTFILE);
366  11     9:0    407 END;
367  11     9:0    426
368  11    10:D      1 PROCEDURE ONEOP;
369  11    10:D      1 [ADJ  FJP  SAS  RNP  CIP  UJP   OM  STM  RBP  CBP  CLP  CGP  EFJ  NFJ]
```

```
370  11   10:0     1
371  11   10:0     1 VAR   JUMPSIZE:INTEGER;
372  11   10:0     2       PCALL:BOOLEAN;
373  11   10:0     3
374  11   11:0     1 PROCEDURE JUMPOPST;
375  11   11:0     1 VAR   NEG:BOOLEAN;
376  11   11:0     0 BEGIN
377  11   11:1     0   NEG:=(JUMPSIZE<0);
378  11   11:1     7   IF NEG THEN JUMPSIZE:=-JUMPSIZE;
379  11   11:1    17   BYTESIZE:=-1;
380  11   11:1    21   REPEAT
381  11   11:2    21     BYTESIZE:=BYTESIZE + 1;
382  11   11:2    26     JUMPSIZE:=JUMPSIZE DIV 2;
383  11   11:1    34   UNTIL JUMPSIZE=0;
384  11   11:1    41   IF NEG THEN
385  11   11:2    44     JUMPSTATS.NEG[BYTESIZE]:=JUMPSTATS.NEG[BYTESIZE] + 1
386  11   11:1    63   ELSE
387  11   11:2    68     JUMPSTATS.POS[BYTESIZE]:=JUMPSTATS.POS[BYTESIZE] + 1;
388  11   11:0    90 END;
389  11   11:0   104
390  11   10:0     0 BEGIN(* ONEOP *)
391  11   10:1     0   WITH OPCODE[BITE]^ DO
392  11   10:2    13     BEGIN
393  11   10:3    13       TOTAL1:=TOTAL1 + 1;
394  11   10:3    19       IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
395  11   10:3    45       IF (BITE=173) OR (BITE=193) THEN DONEPROC:=TRUE;
396  11   10:3    61       IF (BITE IN [161,185,211,212]) THEN
397  11   10:4    84         BEGIN
398  11   10:5    84           BITE:=GETBYTE;
399  11   10:5    95           IF BITE<128 THEN
400  11   10:6   102             BEGIN
401  11   10:7   102               JUMPTOTAL:=JUMPTOTAL + 1;
402  11   10:7   108               JUMPSIZE:=BITE;
403  11   10:7   111               JUMPOPST;
404  11   10:7   113               IF DISPLAY THEN WRITELN(LISTFILE,
405  11   10:8   116                     BUFSTART + BYTEPOS + BITE - PROCSTART:6,' ':18,CODE);
406  11   10:6   164             END
407  11   10:5   164           ELSE
408  11   10:6   166             BEGIN
409  11   10:7   166               JUMPTOTAL:=JUMPTOTAL + 1;
410  11   10:7   172               JUMPSIZE:=JUMPS[(256-BITE-8)DIV 2] - (BUFSTART+BYTEPOS-PROCSTART);
```

```
411  11   10:7   199                JUMPOPST;
412  11   10:7   201                IF DISPLAY THEN WRITELN(LISTFILE,
413  11   10:8   204                                        JUMPSC(256 - BITE - 8) DIV 2]:6,' ':18,CODE);
414  11   10:6   262              END;
415  11   10:4   262            END
416  11   10:3   262          ELSE
417  11   10:4   264            BEGIN
418  11   10:5   264              PCALL:=(BITE IN C174,206,207]);
419  11   10:5   282              BITE:=GETBYTE;
420  11   10:5   293              IF PCALL THEN
421  11   10:6   296                PROCCALLCSEGNUM]^CBITE]:=PROCCALLCSEGNUM]^CBITE] + 1;
422  11   10:5   336              IF DISPLAY THEN WRITELN(LISTFILE,BITE:6,' ':18,CODE);
423  11   10:5   380              IF DONEPROC THEN
424  11   10:6   384                IF DISPLAY THEN WRITELN(LISTFILE);
425  11   10:4   395            END;
426  11   10:3   395          BYTESIZE:=MOSTSIGBIT(BITE);
427  11   10:3   402          BYTEONE1CBYTESIZE]:=BYTEONE1CBYTESIZE] + 1;
428  11   10:2   424        END;
429  11   10:0   424  END;
430  11   10:0   440
431  11   12:D     1  PROCEDURE OPTOP;
432  11   12:D     1  CINC  IND  IXA  LAO  LDO  MOV  MVB  SRO  LLA  LDL  STL  BTP]
433  11   12:D     1  VAR   BIG:INTEGER;
434  11   12:D     2        LOCAL,GLOBAL:BOOLEAN;
435  11   12:0     0  BEGIN
436  11   12:1     0    WITH OPCODECBITE]^ DO
437  11   12:2    13      BEGIN
438  11   12:3    13        TOTAL3:=TOTAL3 + 1;
439  11   12:3    19        IF DATAWATCH THEN
440  11   12:4    23          BEGIN
441  11   12:5    23            LOCAL:=(BITE IN C198,202,204]);
442  11   12:5    41            GLOBAL:=(BITE IN C165,167,171]);
443  11   12:4    59          END;
444  11   12:3    59        IF DISPLAY THEN WRITE(LISTFILE,NAMESCBITE]);
445  11   12:3    85        BIG:=GETBIG;
446  11   12:3    91        BYTESIZE:=MOSTSIGBIT(BIG);
447  11   12:3    98        PARMONE3CBYTESIZE]:=PARMONE3CBYTESIZE] + 1;
448  11   12:3   120        IF DATAWATCH THEN
449  11   12:4   124          IF LOCAL THEN ACTACCESS(LEXLEVEL,BIG)
450  11   12:4   130          ELSE IF GLOBAL THEN ACTACCESS(0,BIG);
451  11   12:3   141        IF DISPLAY THEN WRITELN( STFILE,BIG:6,' ':18,CODE);
```

```
452  11   12:2   185      END;
453  11   12:0   185  END;
454  11   12:0   198
455  11   13:D     1  PROCEDURE LOPTOP;
456  11   13:D     1  [LDA  LOD  STR]
457  11   13:D     1  VAR   BIG,LINKS:INTEGER;
458  11   13:0     0  BEGIN
459  11   13:1     0    WITH OPCODE[BITE]^ DO
460  11   13:2    13      BEGIN
461  11   13:3    13        TOTAL4:=TOTAL4 + 1;
462  11   13:3    19        IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
463  11   13:3    45        BITE:=GETBYTE;
464  11   13:3    56        IF DISPLAY THEN WRITE(LISTFILE,BITE:6);
465  11   13:3    69        LINKS:=BITE;
466  11   13:3    72        BYTESIZE:=MOSTSIGBIT(BITE);
467  11   13:3    79        BYTEONE4[BYTESIZE]:=BYTEONE4[BYTESIZE] + 1;
468  11   13:3   101        BIG:=GETBIG;
469  11   13:3   107        BYTESIZE:=MOSTSIGBIT(BIG);
470  11   13:3   114        PARMTWO4[BYTESIZE]:=PARMTWO4[BYTESIZE] + 1;
471  11   13:3   136        IF DATAWATCH THEN ACTACCESS(LEXLEVEL - LINKS,BIG);
472  11   13:3   147        IF DISPLAY THEN WRITELN(LISTFILE,BIG:6,' ':12,CODE);
473  11   13:2   191      END;
474  11   13:0   191  END;
475  11   13:0   204
476  11   14:D     1  PROCEDURE TWOOP;
477  11   14:D     1  [IXP  CXP]
478  11   14:D     1  VAR   BYTEONE,BYTETWO:BYTE;
479  11   14:D     3        EXTPR:BOOLEAN;
480  11   14:0     0  BEGIN
481  11   14:1     0    WITH OPCODE[BITE]^ DO
482  11   14:2    13      BEGIN
483  11   14:3    13        TOTAL2:=TOTAL2+ 1;
484  11   14:3    19        IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
485  11   14:3    45        IF BITE=205 THEN EXTPR:=TRUE ELSE EXTPR:=FALSE;
486  11   14:3    60        BYTEONE:=GETBYTE;
487  11   14:3    71        BYTESIZE:=MOSTSIGBIT(BYTEONE);
488  11   14:3    78        BYTEONE2[BYTESIZE]:=BYTEONE2[BYTESIZE] + 1;
489  11   14:3   100        BYTETWO:=GETBYTE;
490  11   14:3   111        DONEPROC:=(EXTPR) AND (BYTEONE=0) AND (BYTETWO=2);
491  11   14:3   122        IF (EXTPR) AND (BYTEONE=0) AND (BYTETWO>1) AND (BYTETWO<30) THEN
492  11   14:4   137          BEGIN
```

```
493  11   14:5   137          FLAVOR2[BYTETWO]:=FLAVOR2[BYTETWO] + 1;
494  11   14:5   163          IF DISPLAY THEN WRITELN(LISTFILE,NAMES[56 + BYTETWO],' ':16,CODE);
495  11   14:4   222        END
496  11   14:3   222      ELSE
497  11   14:4   224        BEGIN
498  11   14:5   224          IF EXTPR THEN
499  11   14:6   227            PROCCALL[BYTEONE]^[BYTETWO]:=PROCCALL[BYTEONE]^[BYTETWO] + 1;
500  11   14:5   267          IF DISPLAY THEN WRITELN(LISTFILE,BYTEONE:6,BYTETWO:6,' ':12,CODE);
501  11   14:4   321        END;
502  11   14:3   321      BYTESIZE:=MOSTSIGBIT(BYTETWO);
503  11   14:3   328      BYTETWO2[BYTESIZE]:=BYTETWO2[BYTESIZE] + 1;
504  11   14:2   350    END;
505  11   14:0   350 END;
506  11   14:0   362
507  11   14:0   362
508  11   15:D     1 PROCEDURE WORDOP;
509  11   15:D     1 [  LCI  ]
510  11   15:D     1 VAR   WERD:INTEGER;
511  11   15:0     0 BEGIN
512  11   15:1     0   WITH OPCODE[BITE]^ DO
513  11   15:2    13     BEGIN
514  11   15:3    13       TOTAL3:=TOTAL3+ 1;
515  11   15:3    19       IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
516  11   15:3    45       WERD:=GETWORD;
517  11   15:3    51       IF DISPLAY THEN WRITELN(LISTFILE,WERD:6,' ':18,CODE);
518  11   15:3    95       BYTESIZE:=MOSTSIGBIT(WERD);
519  11   15:3   102       PARMONE3[BYTESIZE]:=PARMONE3[BYTESIZE] + 1;
520  11   15:2   124     END;
521  11   15:0   124 END;
522  11   15:0   136
523  11   16:D     1 PROCEDURE WORDSOP;
524  11   16:D     1 [  XJP  ]
525  11   16:D     1 VAR   WORD1,WORD2,WORD3:INTEGER;
526  11   16:0     0 BEGIN
527  11   16:1     0   WITH OPCODE[BITE]^ DO
528  11   16:2    13     BEGIN
529  11   16:3    13       TOTAL5:=TOTAL5 + 1;
530  11   16:3    19       IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
531  11   16:3    45       IF ODD(BYTEPOS) THEN BITE:=GETBYTE;
532  11   16:3    59       WORD1:=GETWORD;
533  11   16:3    65       BYTESIZE:=MOSTSIGBIT(WOR  );
```

```
534  11   16:3    72        PARMONE5[BYTESIZE]:=PARMONE5[BYTESIZE] + 1;
535  11   16:3    94        WORD2:=GETWORD;
536  11   16:3   100        BYTESIZE:=MOSTSIGBIT(WORD2);
537  11   16:3   107        PARMTWO5[BYTESIZE]:=PARMTWO5[BYTESIZE] + 1;
538  11   16:3   129        BYTESIZE:=MOSTSIGBIT(WORD2-WORD1+1);
539  11   16:3   140        PARMTHREE5[BYTESIZE]:=PARMTHREE5[BYTESIZE] + 1;
540  11   16:3   162        BITE:=GETBYTE;   BITE:=GETBYTE;
541  11   16:3   184        IF BITE<128 THEN
542  11   16:4   191          WORD3:=BUFSTART + BYTEPOS + BITE - PROCSTART
543  11   16:3   196        ELSE
544  11   16:4   203          WORD3:=JUMPS[(256 - BITE - 8) DIV 2];
545  11   16:3   223        IF DISPLAY THEN WRITELN(LISTFILE,WORD1:6,WORD2:6,WORD3:6,' ':6,CODE);
546  11   16:3   287        WORD2:=WORD2 - WORD1 + 1;
547  11   16:3   294        FOR WORD1:=1 TO WORD2 DO
548  11   16:4   305          BEGIN
549  11   16:5   305            HEXCOUNT:=0;
550  11   16:5   308            CODE:='                    ';
551  11   16:5   332            WORD3:=GETWORD;
552  11   16:5   338            WORD3:=BUFSTART + BYTEPOS - WORD3 - 2 - PROCSTART;
553  11   16:5   350            IF DISPLAY THEN WRITELN(LISTFILE,WORD3:41,' ':18,CODE);
554  11   16:4   394          END;
555  11   16:2   401      END;
556  11   16:0   401  END;
557  11   16:0   416
558  11   17:D     1  PROCEDURE CMPRSSOP;
559  11   17:D     1  [  CSP  ]
560  11   17:0     0  BEGIN
561  11   17:1     0    WITH OPCODE[BITE]^ DO
562  11   17:2    13      BEGIN
563  11   17:3    13        TOTAL6:=TOTAL6 + 1;
564  11   17:3    19        IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
565  11   17:3    45        BITE:=GETBYTE;
566  11   17:3    56        IF DISPLAY THEN WRITELN(LISTFILE,NAMES[86 + BITE],' ':16,CODE);
567  11   17:3   115        FLAVOR6[BITE]:=FLAVOR6[BITE] + 1;
568  11   17:2   137      END;
569  11   17:0   137  END;
570  11   17:0   150
571  11   17:0   150
572  11   18:D     1  PROCEDURE CMPRSS2OP;
573  11   18:D     1  [EQU  GEQ  GTR  LEQ  LES  NEQ]
574  11   18:D     1  VAR   BIG:INTEGER;
```

```
575  11    18:0      0 BEGIN
576  11    18:1      0   WITH OPCODE[BITE]^ DO
577  11    18:2     13     BEGIN
578  11    18:3     13       TOTAL7:=TOTAL7 + 1;
579  11    18:3     19       IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
580  11    18:3     45       BITE:=GETBYTE;
581  11    18:3     56       FLAVOR7[BITE DIV 2]:=FLAVOR7[BITE DIV 2] +1;
582  11    18:3     86       IF (BITE=10) OR (BITE=12) THEN BIG:=GETBIG;
583  11    18:3    101       IF DISPLAY THEN
584  11    18:4    104         CASE BITE OF
585  11    18:4    107            2:WRITELN(LISTFILE,'REAL',' ':20,CODE);
586  11    18:4    156            4:WRITELN(LISTFILE,'STR ',' ':20,CODE);
587  11    18:4    205            6:WRITELN(LISTFILE,'BOOL',' ':20,CODE);
588  11    18:4    254            8:WRITELN(LISTFILE,'POWR',' ':20,CODE);
589  11    18:4    303           10:WRITELN(LISTFILE,'BYTE',BIG:6,' ':14,CODE);
590  11    18:4    362           12:WRITELN(LISTFILE,'WORD',BIG:6,' ':14,CODE)
591  11    18:4    419         END;
592  11    18:2    450     END;
593  11    18:0    450 END;
594  11    18:0    468
595  11    19:D      1 PROCEDURE CHRSOP;
596  11    19:D      1 [  LCA  ]
597  11    19:D      1 VAR  SKIPOVER,I:INTEGER;
598  11    19:0      0 BEGIN
599  11    19:1      0   WITH OPCODE[BITE]^ DO
600  11    19:2     13     BEGIN
601  11    19:3     13       TOTAL1:=TOTAL1 + 1;
602  11    19:3     19       IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
603  11    19:3     45       BITE:=GETBYTE;
604  11    19:3     56       IF DISPLAY THEN WRITE(LISTFILE,BITE:6,'   ''');
605  11    19:3     85       BYTESIZE:=MOSTSIGBIT(BITE);
606  11    19:3     92       BYTEONE1[BYTESIZE]:=BYTEONE1[BYTESIZE] + 1;
607  11    19:3    114       IF DISPLAY THEN
608  11    19:4    117         FOR I:=1 TO BITE DO WRITE(LISTFILE,CHR(GETBYTE))
609  11    19:3    141       ELSE
610  11    19:4    150         FOR I:=1 TO BITE DO SKIPOVER:=GETBYTE;
611  11    19:3    174       IF DISPLAY THEN WRITELN(LISTFILE,'''');
612  11    19:2    195     END;
613  11    19:0    195 END;
614  11    19:0    212
615  11    19:0    212
```

```
616  11   20:0     1 PROCEDURE BLKOP;
617  11   20:0     1 {  LOC  }
618  11   20:0     1 VAR  WERD,I,SKIPOVER:INTEGER;
619  11   20:0     0 BEGIN
620  11   20:1     0   WITH OPCODE[BITE]^ DO
621  11   20:2    13     BEGIN
622  11   20:3    13       TOTAL1:=TOTAL1 + 1;
623  11   20:3    19       IF DISPLAY THEN WRITE(LISTFILE,NAMES[BITE]);
624  11   20:3    45       BITE:=GETBYTE;
625  11   20:3    56       IF DISPLAY THEN WRITELN(LISTFILE,BITE:6,' ':18,CODE);
626  11   20:3   100       BYTESIZE:=MOSTSIGBIT(BITE);
627  11   20:3   107       BYTEONE1[BYTESIZE]:=BYTEONE1[BYTESIZE] + 1;
628  11   20:3   129       IF ODD(BYTEPOS) THEN SKIPOVER:=GETBYTE;
629  11   20:3   138       FOR I:=1 TO BITE DO
630  11   20:4   149       BEGIN
631  11   20:5   149         HEXCOUNT:=0;
632  11   20:5   152         CODE:='                    ';
633  11   20:5   176         WERD:=GETWORD;
634  11   20:5   182         IF DISPLAY THEN WRITELN(LISTFILE,WERD:41,' ':18,CODE);
635  11   20:4   226       END;
636  11   20:2   233     END;
637  11   20:0   233 END;
638  11   20:0   248
638  11   20:0   248 (*$I DISASM1.TEXT *)
639  11   20:0   248                          {START OF DISASM1.TEXT}
640  11   20:0   248          {COPYRIGHT (C) REGENTS OF UNIVERSITY OF CALIFORNIA AT SAN DIEGO}
641  11   20:0   248
642  11   21:0     1 PROCEDURE PROCEJUR;
643  11   21:0     1 VAR HEX:HEXTYPE;
644  11   21:0     2     LINENUM,LPROCNUM:INTEGER;
645  11   21:0     4
646  11   22:0     1 PROCEDURE JUMPINFO;
647  11   22:0     1 VAR  OTHERBYTE:INTEGER;
648  11   22:0     0 BEGIN
649  11   22:1     0   BACKJUMP:=0; BYTEPOS:=BYTEPOS - 6; OFFSET:=OFFSET - 6;
650  11   22:1    14   REPEAT
651  11   22:2    14     BACKJUMP:=BACKJUMP + 1;
652  11   22:2    20     OTHERBYTE:=LASTBYTE;
653  11   22:2    26     BITE:=LASTBYTE;
654  11   22:2    37     IF SWAP THEN {JUMPS RELATIVE TO START OF SEGMENT}
655  11   22:3    41       JUMPS[BACKJUMP]:=BUFSTART + BYTEPOS - BITE*256 - OTHERBYTE
```

```
656  11   22:2    60      ELSE
657  11   22:3    65        BEGIN
658  11   22:4    65          JUMPS[BACKJUMP]:=BUFSTART + BYTEPOS - BITE - OTHERBYTE*256;
659  11   22:4    87          BITE:=OTHERBYTE;
660  11   22:3    95        END;
661  11   22:1    95    UNTIL (BITE>127) OR (BACKJUMP=99);
662  11   22:1   105    JUMPS[0]:=BACKJUMP - 1;
663  11   22:1   119    IF BYTEPOS - OFFSET<0 THEN
664  11   22:2   127        BYTEPOS:=BUFRESET(BUFSTART + BYTEPOS,-OFFSET,-1)
665  11   22:1   135      ELSE
666  11   22:2   143        BYTEPOS:=BYTEPOS - OFFSET;
667  11   22:1   149    PROCSTART:=BUFSTART + BYTEPOS; [JUMPS NOW RELATIVE TO START OF PROCEDURE]
668  11   22:1   154    FOR BACKJUMP:=1 TO JUMPS[0] DO
669  11   22:2   175      JUMPS[BACKJUMP]:=JUMPS[BACKJUMP] - PROCSTART;
670  11   22:0   208  END;
671  11   22:0   224
672  11   21:0     0  BEGIN (*PROCEJUR*)
673  11   21:1     0    IF PROCS[PROCNUM]=0 THEN
674  11   21:2    16      WRITELN('PROCEDURE NOT IN FILE')
675  11   21:1    57    ELSE
676  11   21:2    59      BEGIN
677  11   21:3    59        BYTEPOS:=SEGSIZE - BUFSTART - 2*(PROCNUM + 1) - PROCS[PROCNUM] - 2;
678  11   21:3    86        IF BYTEPOS<0 THEN
679  11   21:4    91          BYTEPOS:=BUFRESET(SEGSIZE - 2*(PROCNUM + 1),-PROCS[PROCNUM] - 2,-1)
680  11   21:3   116        ELSE IF BYTEPOS>2556 THEN
681  11   21:5   131          BYTEPOS:=BUFRESET(BUFSTART + BYTEPOS,0,1);
682  11   21:3   142        OFFSET:=GETWORD;                    [ POINTER TO ENTER IC ]
683  11   21:3   148        LPROCNUM:=GETBYTE;
684  11   21:3   154        LEXLEVEL:=GETBYTE;
685  11   21:3   160        BYTEPOS:=BYTEPOS - 4;
686  11   21:3   165        IF LEXLEVEL=255 THEN LEXLEVEL:=-1;
687  11   21:3   177        IF NOT (LEXCHECK OR LEXLOOK) THEN
688  11   21:4   185          IF LPROCNUM=0 THEN
689  11   21:5   190            WRITELN('PROCEDURE ',PROCNUM:3,' IS WRITTEN IN ASSEMBLY.')
690  11   21:4   266          ELSE
691  11   21:5   268          BEGIN
692  11   21:6   268            JUMPINFO;
693  11   21:6   270            DONEPROC:=FALSE;
694  11   21:6   273            IF DISPLAY THEN WRITELN(LISTFILE,
695  11   21:7   276                  ' ':10,'BLOCK #',BYTEPOS DIV 512 + BUFSTBLK:3,
696  11   21:7   322                  '     OFFSET (  BLOCK=',BYTEPOS MOD 512:3,CR,
```

```
697  11   21:7   379                'SEGMENT PROC     OFFSET#',' ':35,'HEX CODE');
698  11   21:6   453              IF NOT CONSOLE THEN
699  11   21:7   458                IF CONTROL THEN WRITE(CR,'[',PROCNUM:2,']')
700  11   21:7   502              ELSE WRITE('.');
701  11   21:6   514              LINENUM:=0;
702  11   21:6   517              REPEAT
703  11   21:7   517                HEX.WORD:=BUFSTART + BYTEPOS - PROCSTART;
704  11   21:7   525                IF DISPLAY THEN WRITE(LISTFILE,SEGNUM:7,PROCNUM:5,HEX.WORD:6,'(',
705  11   21:8   568                    HEXCHAR[HEX.DUM1],HEXCHAR[HEX.HI],HEXCHAR[HEX.LO],'):   ');
706  11   21:7   648                IF CONTROL AND NOT CONSOLE THEN
707  11   21:8   656                  BEGIN
708  11   21:9   656                    WRITE('.');
709  11   21:9   666                    LINENUM:=LINENUM + 1;
710  11   21:9   671                    IF (LINENUM MOD 50=0) THEN WRITE(CR,'    ');
711  11   21:8   704                  END;
712  11   21:7   704                HEXCOUNT:=0;
713  11   21:7   707                CODE:='                ';
714  11   21:7   731                BITE:=GETBYTE;
715  11   21:7   742                OPTOTAL:=OPTOTAL + 1;
716  11   21:7   748                CASE RECTYPES[BITE] OF
717  11   21:7   763                       SHORT:SHORTOP;
718  11   21:7   767                      CMPRSS:CMPRSSOP;
719  11   21:7   771                     CMPRSS2:CMPRSS2OP;
720  11   21:7   775                         ONE:ONEOP;
721  11   21:7   779                        CHRS:CHRSOP;
722  11   21:7   783                         BLK:BLKOP;
723  11   21:7   787                         OPT:OPTOP;
724  11   21:7   791                        LOPT:LOPTOP;
725  11   21:7   795                         TWO:TWOOP;
726  11   21:7   799                       WORDS:WORDSOP;
727  11   21:7   803                        WORD:WORDOP
728  11   21:7   803                  END;
729  11   21:6   836              UNTIL DONEPROC;
730  11   21:5   840            END;
731  11   21:2   840        END;
732  11   21:0   840  END;
733  11   21:0   862
734  11   23:D     1  PROCEDURE ALLPROCS;
735  11   23:D     1  VAR I,J,MAXDIST,INDEX:INTEGER;
736  11   23:D     5      SORTNUMS:ARRAY[0..MAXPROCNUM] OF INTEGER;
737  11   23:D   156      SORTPROCS:ARRAY[0..MAXPROCNUM] OF BYTE;
```

```
738  11    23:0      0 BEGIN
739  11    23:1      0   IF DISPLAY THEN
740  11    23:2      3     BEGIN
741  11    23:3      3       SORTNUMS:=PROCS;
742  11    23:3     11       FOR I:=1 TO MAXPROCNUM DO SORTPROCS[I]:=I;
743  11    23:3     52       FOR I:=1 TO PROCS[0] DO
744  11    23:4     77         BEGIN
745  11    23:5     77           MAXDIST:=0;
746  11    23:5     80           INDEX:=0;
747  11    23:5     83           FOR J:=I TO PROCS[0] DO
748  11    23:6    108             IF SORTNUMS[J]>=MAXDIST THEN
749  11    23:7    123               BEGIN
750  11    23:8    123                 MAXDIST:=SORTNUMS[J];
751  11    23:8    136                 INDEX:=J;
752  11    23:7    139               END;
753  11    23:5    146           SORTNUMS[INDEX]:=SORTNUMS[I];
754  11    23:5    168           SORTNUMS[I]:=SORTPROCS[INDEX];
755  11    23:5    191           SORTPROCS[INDEX]:=SORTPROCS[I];
756  11    23:4    220         END;
757  11    23:3    227       FOR I:=1 TO PROCS[0] DO
758  11    23:4    252         BEGIN
759  11    23:5    252           PROCNUM:=SORTNUMS[I];
760  11    23:5    265           IF (NOT CONSOLE) AND (I MOD 50=0) THEN WRITE(CR,'    ');
761  11    23:5    302           PROCEJUR;
762  11    23:4    304         END;
763  11    23:2    311     END
764  11    23:1    311   ELSE FOR PROCNUM:=1 TO PROCS[0] DO
765  11    23:3    338     BEGIN
766  11    23:4    338       IF (NOT CONSOLE) AND (PROCNUM MOD 50=0) THEN WRITE(CR,'    ');
767  11    23:4    375       PROCEJUR;
768  11    23:3    377     END;
769  11    23:0    384 END;
770  11    23:0    410
771  11    24:D      1 PROCEDURE SEGMINT;
772  11    24:0      0 BEGIN
773  11    24:1      0   IF SWAP THEN
774  11    24:2      4     BEGIN
775  11    24:3      4       SEGSTBLK:=SEGDIREC[SEGNUM*4 + 1];
776  11    24:3     20       SEGSIZE:=SEGDIREC[SEGNUM*4 + 3] + SEGDIREC[SEGNUM*4 + 2]*256;
777  11    24:2     55     END
778  11    24:1     55   ELSE
```

```
779  11    24:2     57      BEGIN
780  11    24:3     57        SEGSTBLK:=SEGDIREC[SEGNUM*4];
781  11    24:3     71        SEGSIZE:=SEGDIREC[SEGNUM*4 + 3]*256 + SEGDIREC[SEGNUM*4 + 2];
782  11    24:2    106      END;
783  11    24:1    106    BUFSTBLK:=SEGSTBLK;
784  11    24:1    110    IF SEGSIZE>2560 THEN
785  11    24:2    116        BYTEPOS:=BUFRESET(SEGSIZE,-1,1)
786  11    24:1    123      ELSE
787  11    24:2    131        BYTEPOS:=BUFRESET(SEGSIZE,-1,0);
788  11    24:1    142    PROCS[0]:=BUFFER[BYTEPOS];  (* NUMBER OF PROCS IN SEGMENT *)
789  11    24:1    164    BYTEPOS:=BYTEPOS - 2*PROCS[0] - 1;
790  11    24:1    184    FOR PROCNUM:=PROCS[0] DOWNTO 1 DO PROCS[PROCNUM]:=GETWORD;
791  11    24:1    229    IF NOT (CONTROL OR LEXCHECK) THEN ALLPROCS;
792  11    24:0    239 END;
793  11    24:0    254
794  11     8:D      1 PROCEDURE ACTACCESS; [FINALEX,OFFSET:INTEGER;]
795  11     8:D      3 VAR  FINALPROC,FINALSEG:INTEGER;
796  11     8:D      5      INSIDE:BOOLEAN;
797  11     8:0      0 BEGIN
798  11     8:1      0   IF (FINALEX=PROCLEX[DATAPROC]) AND (PROCNUM>=DATAPROC) THEN
799  11     8:2     20     IF SEGNUM=DATASEG THEN
800  11     8:3     26       BEGIN
801  11     8:4     26         INSIDE:=(PROCNUM=DATAPROC);
802  11     8:4     32         FINALPROC:=PROCNUM;
803  11     8:4     35         WHILE PROCLEX[FINALPROC]>PROCLEX[DATAPROC] DO FINALPROC:=FINALPROC - 1;
804  11     8:4     66         IF FINALPROC=DATAPROC THEN
805  11     8:4     72           [$R-]
806  11     8:5     72           DSSTART^[OFFSET]:=DSSTART^[OFFSET] + 1;
807  11     8:5     84           [$R+]
808  11     8:3     84       END
809  11     8:2     84     ELSE IF (DATAPROC=1) AND (SEGNUM>DATASEG) THEN
810  11     8:4     97       BEGIN
811  11     8:5     97         FINALSEG:=SEGNUM;
812  11     8:5    100         WHILE SEGLEX[FINALSEG]>SEGLEX[DATASEG] DO FINALSEG:=FINALSEG - 1;
813  11     8:5    131         IF FINALSEG=DATASEG THEN
814  11     8:5    137           [$R-]
815  11     8:6    137           DSSTART^[OFFSET]:=DSSTART^[OFFSET] + 1;
816  11     8:6    149           [$R+]
817  11     8:4    149       END;
818  11     8:0    149 END;
819  11     8:0    168
```

```
820  11   25:0     1 PROCEDURE PROCGUIDE;
821  11   25:0     1 TYPE  SPACEPTR=^SPACE;
822  11   25:0     1       SPACE=ARRAY[0..19] OF INTEGER;
823  11   25:0     1 VAR   I,J:INTEGER;
824  11   25:0     3       DSSPACE:SPACEPTR;
825  11   25:0     4
826  11   26:0     1 PROCEDURE DATASEGINFO;
827  11   26:0     1 VAR  TEMP:INTEGER;
828  11   26:0     0 BEGIN
829  11   26:1     0   PROCEJUR;
830  11   26:1     2   BYTEPOS:=BYTEPOS - 2;
831  11   26:1     7   IF SWAP THEN
832  11   26:2    11     BEGIN
833  11   26:3    11       DTSGSZ:=LASTBYTE;
834  11   26:3    17       DTSGSZ:=DTSGSZ + LASTBYTE*256;
835  11   26:3    30       TEMP:=LASTBYTE;
836  11   26:3    36       DTSGSZ:=DTSGSZ + LASTBYTE*256 + TEMP;
837  11   26:2    51     END
838  11   26:1    51   ELSE
839  11   26:2    53     BEGIN
840  11   26:3    53       DTSGSZ:=LASTBYTE*256;
841  11   26:3    63       DTSGSZ:=DTSGSZ + LASTBYTE;
842  11   26:3    72       TEMP:=LASTBYTE*256;
843  11   26:3    82       DTSGSZ:=DTSGSZ + LASTBYTE + TEMP;
844  11   26:2    93     END;
845  11   26:1    93   DTSGSZ:=DTSGSZ DIV 2;
846  11   26:0    99 END;
847  11   26:0   112
848  11   27:0     1 PROCEDURE PROCLOOK;
849  11   27:0     0 BEGIN
850  11   27:1     0   GOTOXY(0,3); WRITE(' ':50); GOTOXY(0,3);
851  11   27:1    20   LEXLOOK:=TRUE;
852  11   27:1    23   I:=(PROCS[0] DIV 5) + 1;
853  11   27:1    42   FOR J:=0 TO ((PROCS[0]-1) DIV I) DO WRITE('    #  LL  SIZE');
854  11   27:1   110   WRITELN;
855  11   27:1   118   FOR PROCNUM:=1 TO PROCS[0] DO
856  11   27:2   140     BEGIN
857  11   27:3   140       DATASEGINFO;
858  11   27:3   142       GOTOXY(15*((PROCNUM-1) DIV I),5+((PROCNUM-1) MOD I));
859  11   27:3   163       WRITE(PROCNUM:5,':',LEXLEVEL:3,DTSGSZ:6);
860  11   27:2   205     END;
```

```
861  11   27:1   212   FOR J:=1 TO (5 - (PROCS[0] MOD 5)) DO WRITELN;
862  11   27:1   259   PROMPT;
863  11   27:1   262   LEXLOOK:=FALSE;
864  11   27:0   265 END;
865  11   27:0   284
866  11   25:0     0 BEGIN [PROCGUIDE]
867  11   25:1     0   SEGMINT;
868  11   25:1     2   REPEAT
869  11   25:2     2     PAGE(OUTPUT);
870  11   25:2    12     WRITE('PROCEDURE GUIDE:  #(OF PROCEDURE),');
871  11   25:2    58     IF LEXCHECK THEN
872  11   25:3    62       WRITELN('L(ISTING),Q(UIT)')
873  11   25:2    98     ELSE
874  11   25:3   100       WRITELN('A(LL),L(ISTING),Q(UIT)');
875  11   25:2   142     WRITE('  TO SEGMENT: ');
876  11   25:2   168     FOR I:=1 TO 8 DO WRITE(CHR(SEGDIREC[63 + SEGNUM*8 + I]));
877  11   25:2   211     PROCNUM:=0;
878  11   25:2   214     WRITE(CR,CR,'WHICH PROCEDURE ');
879  11   25:2   262     IF LEXCHECK THEN
880  11   25:3   266       WRITE('DATA SEGMENT TO WATCH?')
881  11   25:2   300     ELSE
882  11   25:3   302       WRITE('TO DIS-ASSEMBLE?');
883  11   25:2   330     READ(CH);
884  11   25:2   340     IF (CH='L') OR (CH='L') THEN
885  11   25:3   349         PROCLOOK
886  11   25:2   349     ELSE IF ((CH='A') OR (CH='A')) AND (NOT LEXCHECK) THEN
887  11   25:4   366       BEGIN
888  11   25:5   366         PAGE(OUTPUT);
889  11   25:5   376         WRITELN('DIS-ASSEMBLING ALL',PROCS[0]:3,' PROCEDURES',CR,CR);
890  11   25:5   478         IF NOT DISPLAY THEN WRITE(CR,CR,'(',SEGNUM:2,')');
891  11   25:5   532         ALLPROCS;
892  11   25:5   534         PROMPT;
893  11   25:5   537         CH:='Q';
894  11   25:4   540       END
895  11   25:3   540     ELSE IF (CH>='0') AND (CH<='9') THEN
896  11   25:5   551       BEGIN
897  11   25:6   551         PROCNUM:=ORD(CH)-ORD('0');
898  11   25:6   556         READ(CH);
899  11   25:6   566         IF (CH>='0') AND (CH<='9') THEN
900  11   25:7   575           PROCNUM:=PROCNUM*10 + ORD(CH) - ORD('0');
901  11   25:6   584         IF (PROCNUM<1) OR (PROCNUM>PROCS[0]) THEN
```

```
902  11   25:7   604            BEGIN
903  11   25:8   604              WRITELN(CR,'I DIDN''T SAY YOU HAD THAT PROCEDURE!');
904  11   25:8   670              PROMPT;
905  11   25:7   673            END
906  11   25:6   673          ELSE IF NOT LEXCHECK THEN
907  11   25:8   680            BEGIN
908  11   25:9   680              PAGE(OUTPUT);
909  11   25:9   690              WRITELN('DIS-ASSEMBLING PROCEDURE',PROCNUM:3,CR);
910  11   25:9   754              PROCEJUR;
911  11   25:9   756              PROMPT;
912  11   25:9   759              CH:=' ';
913  11   25:8   762            END
914  11   25:7   762          ELSE
915  11   25:8   764            BEGIN
916  11   25:9   764              DATAPROC:=PROCNUM;
917  11   25:9   767              DATASEG:=SEGNUM;
918  11   25:9   770              DATASEGINFO;
919  11   25:9   772              DATASEGSIZE:=DTSGSZ;
920  11   25:9   776              NEW(DSSTART);
921  11   25:9   781              FOR I:=1 TO ((DATASEGSIZE+19) DIV 20) DO NEW(DSSPACE);
922  11   25:9   809              FILLCHAR(DSSTART^,DATASEGSIZE*2,0);
923  11   25:9   818              FOR PROCNUM:=1 TO PROCSE[0] DO
924  11   25:0   840                BEGIN
925  11   25:1   840                  PROCEJUR;
926  11   25:1   842                  PROCLEX[PROCNUM]:=LEXLEVEL;
927  11   25:0   854                END;
928  11   25:9   861              CH:=CHR(7);
929  11   25:8   864            END;
930  11   25:5   864        END;
931  11   25:1   864    UNTIL (CH='Q') OR (CH='Q') OR (CH=CHR(7));
932  11   25:0   877  END;
933  11   25:0   908
934  11   28:0     1  PROCEDURE SEGMTGUIDE;
935  11   28:0     1  VAR I,J:INTEGER;
936  11   28:0     0  BEGIN
937  11   28:1     0    REPEAT
938  11   28:2     0      PAGE(OUTPUT);
939  11   28:2    10      WRITELN('SEGMENT GUIDE:  #(OF SEGMENT),Q(UIT)');
940  11   28:2    66      WRITELN(CR,CR,'YOU HAVE THESE SEGMENTS:');
941  11   28:2   130      FOR I:=0 TO 15 DO
942  11   28:3   141        BEGIN
```

```
943  11   28:4   141          WRITE(I:4,'      ');
944  11   28:4   169          FOR J:=1 TO 8 DO WRITE(CHR(SEGDIREC[63 + I*8 + J]));
945  11   28:4   212          WRITELN;
946  11   28:3   220        END;
947  11   28:2   227      WRITE(CR,'WHICH SEGMENT TO LOOK AT ');
948  11   28:2   274      IF LEXCHECK THEN
949  11   28:3   278        WRITE('TO DECIDE ON DATA SEGMENT?')
950  11   28:2   316      ELSE
951  11   28:3   318        WRITE('FOR POSSIBLE DIS-ASSEMBLY?');
952  11   28:2   356      READ(CH);
953  11   28:2   366      IF (CH<>'Q') AND (CH<>'Q') THEN
954  11   28:3   375        BEGIN
955  11   28:4   375          SEGNUM:=0;
956  11   28:4   378          IF (CH>='0') AND (CH<='9') THEN SEGNUM:=ORD(CH)-ORD('0');
957  11   28:4   392          READ(CH);
958  11   28:4   402          IF (CH>='0') AND (CH<='9') THEN
959  11   28:5   411            SEGNUM:=SEGNUM*10 + ORD(CH) - ORD('0');
960  11   28:4   420          IF (SEGDIREC[4*SEGNUM] + SEGDIREC[4*SEGNUM + 1]=0) OR (SEGNUM>15) THEN
961  11   28:5   455            BEGIN
962  11   28:6   455              WRITELN(CR,'I DIDN''T SAY YOU HAD THAT SEGMENT!');
963  11   28:6   519              READ(KEYBOARD,CH);
964  11   28:5   529            END
965  11   28:4   529          ELSE
966  11   28:5   531            BEGIN
967  11   28:6   531              PROCGUIDE;
968  11   28:6   533              IF CH<>CHR(7) THEN CH:='A';
969  11   28:5   541            END;
970  11   28:3   541        END;
971  11   28:1   541    UNTIL (CH='Q') OR (CH='Q') OR (CH=CHR(7));
972  11   28:0   554  END;
973  11   28:0   574
974  11   29:0     1  PROCEDURE LEXGUIDE;
975  11   29:0     0  BEGIN
976  11   29:1     0    LEXCHECK:=TRUE;
977  11   29:1     3    DATASEG:=-1;
978  11   29:1     7    REPEAT
979  11   29:2     7      SEGMTGUIDE;
980  11   29:2     9      IF (CH='Q') OR (CH='Q') THEN
981  11   29:3    18        BEGIN
982  11   29:4    18          PAGE(OUTPUT);
983  11   29:4    29          GOTOXY(0,10);
```

```
 984  11   29:4    33          WRITELN('HAVE YOU CHANGED YOUR MIND ABOUT DATA SEGMENT WATCHING?');
 985  11   29:4   103          READ(KEYBOARD,CH);
 986  11   29:4   118          IF (CH='Y') OR (CH='Y') THEN DATAWATCH:=FALSE;
 987  11   29:3   130        END;
 988  11   29:1   130    UNTIL (CH=CHR(7)) OR (NOT DATAWATCH);
 989  11   29:1   139    IF DATAWATCH THEN
 990  11   29:2   143      FOR SEGNUM:=0 TO 15 DO
 991  11   29:3   154        IF SEGDIREC[4*SEGNUM] + SEGDIREC[4*SEGNUM + 1]<>0 THEN
 992  11   29:4   185          BEGIN
 993  11   29:5   185            SEGMINT; [SETS UP APPROPIATE SEGMENT]
 994  11   29:5   187            PROCNUM:=1;
 995  11   29:5   190            PROCEJUR; [SETS UP PROCEDURE TO DETERMINE SEGMENT'S LEXLEVEL]
 996  11   29:5   192            SEGLEX[SEGNUM]:=LEXLEVEL;
 997  11   29:4   204          END
 998  11   29:3   204        ELSE SEGLEX[SEGNUM]:=100;
 999  11   29:1   224    PAGE(OUTPUT);
1000  11   29:1   234    LEXCHECK:=FALSE;
1001  11   29:0   237  END;
1002  11   29:0   254
1003  11    1:0     0  BEGIN (* SEGMENT DISASSEMBLE *)
1004  11    1:1     0    PAGE(OUTPUT);
1005  11    1:1    10    GOTOXY(0,10);
1006  11    1:1    15    WRITE('
1007  11    1:1    86                 ',          DO YOU WISH TO KEEP TRACK OF REFERENCES',CR,
1008  11    1:1   148                               TO A PARTICULAR PROCEDURE''S DATA SEGMENT?');
1009  11    1:1   158    READ(KEYBOARD,CH);
1010  11    1:1   167    DATAWATCH:=(CH='Y') OR (CH='Y');
1011  11    1:1   178    IF DATAWATCH THEN LEXGUIDE ELSE LEXCHECK:=FALSE;
1012  11    1:1   188    PAGE(OUTPUT);
1013  11    1:1   193    GOTOXY(0,10);
1014  11    1:1   243    WRITE('DO YOU WISH CONTROL OVER DIS-ASSEMBLY?');
1015  11    1:1   253    READ(KEYBOARD,CH);
1016  11    1:1   262    CONTROL:=(CH='Y') OR (CH='Y');
1017  11    1:2   266    IF CONTROL THEN
1018  11    1:3   266      BEGIN
1019  11    1:3   276        PAGE(OUTPUT);
1020  11    1:3   281        GOTOXY(0,7);
1021  11    1:3   291        WRITE(CHR(7));
1022  11    1:3   362        WRITE('***  WARNING  - -  STATISTICS ARE GATHERED ON DIS-ASSEMBLED');
1023  11    1:3   403        WRITELN(' PROCEDURES ONLY  ***');
1024  11    1:4   453        IF DATAWATCH THEN WRITELN(CR,CR,'                    ',
                                                    '***   THIS INCLUDES DATA SEGMENT WATCHING   **  ');
```

```
1025  11     1:3    520        READ(KEYBOARD,CH);
1026  11     1:3    530        SEGMTGUIDE;
1027  11     1:2    532      END
1028  11     1:1    532    ELSE
1029  11     1:2    534      BEGIN
1030  11     1:3    534        IF NOT CONSOLE THEN WRITE(CHR(12),CR);
1031  11     1:3    559        FOR SEGNUM:=0 TO 15 DO
1032  11     1:4    570          BEGIN
1033  11     1:5    570            IF NOT CONSOLE THEN WRITE(CR,'(',SEGNUM:2,')');
1034  11     1:5    615            IF SEGDIREC[4*SEGNUM] + SEGDIREC[4*SEGNUM + 1]<>0 THEN SEGMINT;
1035  11     1:4    648          END;
1036  11     1:3    655        PROMPT;
1037  11     1:2    658      END;
1038  11     1:0    658  END;
1039  11     1:0    674  (*$I DISASM1.TEXT *)
1039  11     1:0    674  (*$I DISASM2.TEXT*)
1040  11     1:0    674
1041  11     1:0    674                              [START OF DISASM2.TEXT]
1042  11     1:0    674          [COPYRIGHT (C) REGENTS OF UNIVERSITY OF CALIFORNIA AT SAN DIEGO]
1043  11     1:0    674
1044  12     1:D      1  SEGMENT PROCEDURE GATHER;
1045  12     1:D      1  VAR   FILENAME:STRING;
1046  12     1:D     42
1047  12     2:D      1  PROCEDURE WRITEHDR(VAR H:INTERACTIVE;HEADER:INTEGER);
1048  12     2:0      0  BEGIN
1049  12     2:1      0    CASE HEADER OF
1050  12     2:1      3      1: WRITELN(H,'          PARAMETER ONE');
1051  12     2:1     43      2: WRITELN(H,'BITS USED    TOTAL    PERCENTAGE');
1052  12     2:1     93      3: WRITELN(H,'          PARAMETER ONE            PARAMETER TWO          ');
1053  12     2:1    166      4: WRITELN(H,'BITS USED    TOTAL    PERCENTAGE    TOTAL    PERCENTAGE');
1054  12     2:1    239      5: WRITELN(H,'          PARAMETER ONE            PARAMETER TWO',
1055  12     2:2    295                                                    '           CASE TABLE SIZE');
1056  12     2:1    339      6: WRITELN(H,'BITS USED    TOTAL    PERCENTAGE    TOTAL    PERCENTAGE',
1057  12     2:2    404                                                    '    TOTAL    PERCENTAGE');
1058  12     2:1    445      7: WRITELN(H,'FLAVOR      TOTAL    PERCENTAGE     FLAVOR',
1059  12     2:2    497                                                    '      TOTAL    PERCENTAGE');
1060  12     2:1    540      8: WRITELN(H,'   #   TOTAL    PCT   #   TOTAL    PCT   #   TOTAL',
1061  12     2:2    600                                                    '    PCT   #   TOTAL    PCT')
1062  12     2:1    642    END;
1063  12     2:0    668  END;
1064  12     2:0    684
```

```
1065  12    3:D     1 PROCEDURE JUMPSTUFF;
1066  12    3:D     1 VAR   I:INTEGER;
1067  12    3:0     0 BEGIN
1068  12    3:1     0   WRITELN(LISTFILE,CR,'JUMP STATISTICS ON THE',JUMPTOTAL:5,' TOTAL JUMPS');
1069  12    3:1    87   IF JUMPTOTAL>0 THEN
1070  12    3:2    93     BEGIN
1071  12    3:3    93       WRITELN(LISTFILE,CR,
1072  12    3:3   103                '                                POSITIVE JUMPS          NEGATIVE JUMPS');
1073  12    3:3   176       WRITEHDR(LISTFILE,4);
1074  12    3:3   182       WITH JUMPSTATS DO
1075  12    3:4   182         FOR I:=0 TO 15 DO
1076  12    3:5   193           WRITELN(LISTFILE,I+1:5,POS[I]:13,POS[I]/JUMPTOTAL*100:14:2,
1077  12    3:5   251                                NEG[I]:9,NEG[I]/JUMPTOTAL*100:14:2);
1078  12    3:2   312     END
1079  12    3:1   312   ELSE WRITELN(LISTFILE,CR,'SORRY NO JUMPS TODAY!');
1080  12    3:0   365 END;
1081  12    3:0   382
1082  12    5:D     1 PROCEDURE PROCSTUFF;
1083  12    5:D     1 VAR   I,J:INTEGER;
1084  12    5:0     0 BEGIN
1085  12    5:1     0   WRITELN(LISTFILE,CR,'PROCEDURE CALL STATISTICS');
1086  12    5:1    55   FOR I:=0 TO 15 DO
1087  12    5:2    66     IF PROCCALL[I]<>NIL THEN
1088  12    5:3    80       FOR J:=1 TO MAXPROCNUM DO
1089  12    5:4    93         IF PROCCALL[I]^[J]>0 THEN
1090  12    5:5   116           WRITELN(LISTFILE,'  SEGMENT:',I:4,'  PROCEDURE:',J:4,
1091  12    5:5   182                                           '  CALLS:',PROCCALL[I]^[J]:4);
1092  12    5:0   252 END;
1093  12    5:0   274
1094  12    6:D     1 PROCEDURE HISTOGRAM(PCTMAX:INTEGER);
1095  12    6:D     2 VAR  I:INTEGER;
1096  12    6:0     0 BEGIN
1097  12    6:1     0   PCTMAX:=ROUND(PCTMAX/MAXOP*20);
1098  12    6:1    12   FOR I:=1 TO PCTMAX DO WRITE(LISTFILE,'*');
1099  12    6:0    40 END;
1100  12    6:0    54
1101  12    7:D     1 PROCEDURE SHORTSTUFF;
1102  12    7:D     1 VAR  I:INTEGER;
1103  12    7:D     2
1104  12    8:D     1 PROCEDURE SHORT1(VAR H:INTERACTIVE);
1105  12    8:0     0 BEGIN
```

```
1106  12     8:1      0   WRITE(H,CR,'SLDC   OPCODE: 0..127   TOTAL:',
1107  12     8:1     48              SLDC:8,SLDC/OPTOTAL*100:16:2,' % ');
1108  12     8:1     87   HISTOGRAM(SLDC);
1109  12     8:1     91   IF SLDC<>0 THEN
1110  12     8:2     97     BEGIN
1111  12     8:3     97       WRITELN(H,CR); WRITEHDR(H,8);
1112  12     8:3    115       FOR OP:=0 TO 31 DO
1113  12     8:4    126         WRITELN(H,OP:4,':',OPCODE[OP]^.TOTAL0:7,OPCODE[OP]^.TOTAL0/SLDC*100:7:2,
1114  12     8:4    188          OP+32:4,':',OPCODE[OP+32]^.TOTAL0:7,OPCODE[OP+32]^.TOTAL0/SLDC*100:7:2,
1115  12     8:4    256          OP+64:4,':',OPCODE[OP+64]^.TOTAL0:7,OPCODE[OP+64]^.TOTAL0/SLDC*100:7:2,
1116  12     8:4    324          OP+96:4,':',OPCODE[OP+96]^.TOTAL0:7,OPCODE[OP+96]^.TOTAL0/SLDC*100:7:2)
1117  12     8:2    405     END;
1118  12     8:1    405   WRITE(H,CR,CR,'SLDL  OPCODE: 216..231  TOTAL:',
1119  12     8:1    461                 SLDL:8,SLDL/OPTOTAL*100:16:2,' % ');
1120  12     8:1    500   HISTOGRAM(SLDL);
1121  12     8:1    504   IF SLDL<>0 THEN
1122  12     8:2    510     BEGIN
1123  12     8:3    510       WRITELN(H,CR); WRITEHDR(H,8);
1124  12     8:3    528       FOR OP:=216 TO 219 DO
1125  12     8:4    543         WRITELN(H,OP:4,':',OPCODE[OP]^.TOTAL0:7,OPCODE[OP]^.TOTAL0/SLDL*100:7:2,
1126  12     8:4    605          OP+4:4,':',OPCODE[OP+4]^.TOTAL0:7,OPCODE[OP+4]^.TOTAL0/SLDL*100:7:2,
1127  12     8:4    673          OP+8:4,':',OPCODE[OP+8]^.TOTAL0:7,OPCODE[OP+8]^.TOTAL0/SLDL*100:7:2,
1128  12     8:4    741          OP+12:4,':',OPCODE[OP+12]^.TOTAL0:7,OPCODE[OP+12]^.TOTAL0/SLDL*100:7:2)
1129  12     8:2    822     END;
1130  12     8:0    822 END;
1131  12     8:0    846
1132  12     9:0      1 PROCEDURE SHORT2(VAR H:INTERACTIVE);
1133  12     9:0      0 BEGIN
1134  12     9:1      0   WRITE(H,CR,CR,'SLDO  OPCODE: 232..247  TOTAL:',
1135  12     9:1     56                 SLDO:8,SLDO/OPTOTAL*100:16:2,' % ');
1136  12     9:1     95   HISTOGRAM(SLDO);
1137  12     9:1     99   IF SLDO<>0 THEN
1138  12     9:2    105     BEGIN
1139  12     9:3    105       WRITELN(H,CR); WRITEHDR(H,8);
1140  12     9:3    123       FOR OP:=232 TO 235 DO
1141  12     9:4    138         WRITELN(H,OP:4,':',OPCODE[OP]^.TOTAL0:7,OPCODE[OP]^.TOTAL0/SLDO*100:7:2,
1142  12     9:4    200            OP+4:4,':',OPCODE[OP+4]^.TOTAL0:7,OPCODE[OP+4]^.TOTAL0/SLDO*100:7:2,
1143  12     9:4    268            OP+8:4,':',OPCODE[OP+8]^.TOTAL0:7,OPCODE[OP+8]^.TOTAL0/SLDO*100:7:2,
1144  12     9:4    336            OP+12:4,':',OPCODE[OP+12]^.TOTAL0:7,OPCODE[OP+12]^.TOTAL0/SLDO*100:7:
1145  12     9:2    417     END;
1146  12     9:1    417   WRITE(H,CR,CR,'SIND  OPCODE: 248..255  TOTAL:',
```

```
1147  12    9:1   473                    SIND:8,SIND/OPTOTAL*100:16:2,' % ');
1148  12    9:1   512    HISTOGRAM(SIND);
1149  12    9:1   516    IF SIND<>0 THEN
1150  12    9:2   522      BEGIN
1151  12    9:3   522        WRITELN(H,CR); WRITEHDR(H,8);
1152  12    9:3   540        FOR OP:=248 TO 249 DO
1153  12    9:4   555          WRITELN(H,OP:4,':',OPCODE[OP]^.TOTAL0:7,OPCODE[OP]^.TOTAL0/SIND*100:7:2,
1154  12    9:4   617               OP+2:4,':',OPCODE[OP+2]^.TOTAL0:7,OPCODE[OP+2]^.TOTAL0/SIND*100:7:2,
1155  12    9:4   685               OP+4:4,':',OPCODE[OP+4]^.TOTAL0:7,OPCODE[OP+4]^.TOTAL0/SIND*100:7:2,
1156  12    9:4   753               OP+6:4,':',OPCODE[OP+6]^.TOTAL0:7,OPCODE[OP+6]^.TOTAL0/SIND*100:7:2);
1157  12    9:2   834      END;
1158  12    9:1   834    WRITELN(H);
1159  12    9:0   840  END;
1160  12    9:0   864
1161  12    7:0     0  BEGIN(* SHORTSTUFF *)
1162  12    7:1     0    SHORT1(LISTFILE);
1163  12    7:1     5    SHORT2(LISTFILE);
1164  12    7:0    10  END;
1165  12    7:0    22
1166  12   10:D     1  PROCEDURE SHORTST;
1167  12   10:D     1  VAR  I:INTEGER;
1168  12   10:0     0  BEGIN
1169  12   10:1     0    INUM:=OPCODE[OP]^.TOTAL0;
1170  12   10:1    14    WRITE(LISTFILE,INUM:8,INUM/OPTOTAL*100:16:2,' % ');
1171  12   10:1    57    HISTOGRAM(INUM);
1172  12   10:1    60    WRITELN(LISTFILE);
1173  12   10:0    68  END;
1174  12   10:0    80
1175  12   11:D     1  PROCEDURE ONEST;
1176  12   11:D     1  VAR   I:INTEGER;
1177  12   11:0     0  BEGIN
1178  12   11:1     0    WITH OPCODE[OP]^ DO
1179  12   11:2    13      BEGIN
1180  12   11:3    13        INUM:=TOTAL1;
1181  12   11:3    17        WRITE(LISTFILE,INUM:8,INUM/OPTOTAL*100:16:2,' % ');
1182  12   11:3    60        IF TOTAL1<>0 THEN
1183  12   11:4    66          BEGIN
1184  12   11:5    66            HISTOGRAM(TOTAL1);
1185  12   11:5    70            WRITELN(LISTFILE,CR);
1186  12   11:5    88            WRITEHDR(LISTFILE,1); WRITELN(LISTFILE);
1187  12   11:5   102            WRITEHDR(LISTFILE,2);
```

```
1138  12    11:5    108            FOR I:=0 TO 7 DO
1189  12    11:6    119             WRITELN(LISTFILE,I+1:5,BYTEONE1[I]:13,BYTEONE1[I]/TOTAL1*100:14:2);
1190  12    11:4    192          END
1191  12    11:3    192        ELSE WRITELN(LISTFILE);
1192  12    11:2    202      END;
1193  12    11:0    202    END;
1194  12    11:0    218
1195  12    12:D      1  PROCEDURE TWOST;
1196  12    12:D      1  VAR   I:INTEGER;
1197  12    12:0      0  BEGIN
1198  12    12:1      0    WITH OPCODE[OP]^ DO
1199  12    12:2     13      BEGIN
1200  12    12:3     13        WRITE(LISTFILE,TOTAL2:8,TOTAL2/OPTOTAL*100:16:2,' % ');
1201  12    12:3     58        HISTOGRAM(TOTAL2);
1202  12    12:3     62        WRITELN(LISTFILE,CR); WRITEHDR(LISTFILE,3);
1203  12    12:3     86        WRITELN(LISTFILE); WRITEHDR(LISTFILE,4);
1204  12    12:3    100        IF TOTAL2=0 THEN
1205  12    12:4    106          FOR I:=0 TO 7 DO
1206  12    12:5    117            WRITELN(LISTFILE,I+1:5,BYTEONE2[I]:13,0.0:14:2,BYTETWO2[I]:9,0.0:14:2)
1207  12    12:3    206        ELSE
1208  12    12:4    215          FOR I:=0 TO 7 DO
1209  12    12:5    226            WRITELN(LISTFILE,I+1:5,BYTEONE2[I]:13,BYTEONE2[I]/TOTAL2*100:14:2,
1210  12    12:5    284                    BYTETWO2[I]:9,BYTETWO2[I]/TOTAL2*100:14:2);
1211  12    12:3    345        IF OP=205 THEN
1212  12    12:4    352          BEGIN
1213  12    12:5    352            WRITELN(LISTFILE); WRITEHDR(LISTFILE,7);
1214  12    12:5    366            IF TOTAL2=0 THEN
1215  12    12:6    372              FOR I:=2 TO 15 DO
1216  12    12:7    383                WRITELN(LISTFILE,NAMES[56+I],FLAVOR2[I]:9,0.0:14:2,'     ',
1217  12    12:7    461                        NAMES[56+I+14],FLAVOR2[I+14]:9,0.0:14:2)
1218  12    12:5    534            ELSE
1219  12    12:6    543              FOR I:=2 TO 15 DO
1220  12    12:7    554                WRITELN(LISTFILE,NAMES[56+I],FLAVOR2[I]:9,
1221  12    12:7    600                        FLAVOR2[I]/TOTAL2*100:14:2,'     ',
1222  12    12:7    646                        NAMES[56+I+14],FLAVOR2[I+14]:9,
1223  12    12:7    696                        FLAVOR2[I+14]/TOTAL2*100:14:2);
1224  12    12:4    742          END;
1225  12    12:2    742      END;
1226  12    12:0    742  END;
1227  12    12:0    774
1228  12    13:D      1  PROCEDURE WORDST;
```

```
1229  12   13:D    1 VAR   I:INTEGER;
1230  12   13:0    0 BEGIN
1231  12   13:1    0   WITH OPCODE[OP]^ DO
1232  12   13:2   13     BEGIN
1233  12   13:3   13       INUM:=TOTAL3;
1234  12   13:3   17       WRITE(LISTFILE,INUM:8,INUM/OPTOTAL*100:16:2,' % ');
1235  12   13:3   60       IF TOTAL3<>0 THEN
1236  12   13:4   66         BEGIN
1237  12   13:5   66           HISTOGRAM(TOTAL3);
1238  12   13:5   70           WRITELN(LISTFILE,CR); WRITEHDR(LISTFILE,1);
1239  12   13:5   94           WRITELN(LISTFILE); WRITEHDR(LISTFILE,2);
1240  12   13:5  108           FOR I:=0 TO 15 DO
1241  12   13:6  119             WRITELN(LISTFILE,I+1:5,PARMONE3[I]:13,PARMONE3[I]/TOTAL3*100:14:2);
1242  12   13:4  192         END
1243  12   13:3  192       ELSE WRITELN(LISTFILE);
1244  12   13:2  202     END;
1245  12   13:0  202 END;
1246  12   13:0  218
1247  12   14:D    1 PROCEDURE LOPTST;
1248  12   14:D    1 VAR   I:INTEGER;
1249  12   14:0    0 BEGIN
1250  12   14:1    0   WITH OPCODE[OP]^ DO
1251  12   14:2   13     BEGIN
1252  12   14:3   13       INUM:=TOTAL4;
1253  12   14:3   17       WRITE(LISTFILE,INUM:8,INUM/OPTOTAL*100:16:2,' % ');
1254  12   14:3   60       IF TOTAL4<>0 THEN
1255  12   14:4   66         BEGIN
1256  12   14:5   66           HISTOGRAM(TOTAL4);
1257  12   14:5   70           WRITELN(LISTFILE,CR); WRITEHDR(LISTFILE,3);
1258  12   14:5   94           WRITELN(LISTFILE); WRITEHDR(LISTFILE,4);
1259  12   14:5  108           FOR I:=0 TO 7 DO
1260  12   14:6  119             WRITELN(LISTFILE,I+1:5,BYTEONE4[I]:13,BYTEONE4[I]/TOTAL4*100:14:2,
1261  12   14:6  177               PARMTWO4[I]:9,PARMTWO4[I]/TOTAL4*100:14:2);
1262  12   14:5  238           FOR I:=8 TO 15 DO
1263  12   14:6  249             WRITELN(LISTFILE,I+1:5,PARMTWO4[I]:36,PARMTWO4[I]/TOTAL4*100:14:2);
1264  12   14:4  322         END
1265  12   14:3  322       ELSE WRITELN(LISTFILE);
1266  12   14:2  332     END;
1267  12   14:0  332 END;
1268  12   14:0  350
1269  12   15:D    1 PROCEDURE WORDSST;
```

```
1270  12   15:0     1 VAR   I:INTEGER;
1271  12   15:0     0 BEGIN
1272  12   15:1     0   WITH OPCODE[OP]^ DO
1273  12   15:2    13     BEGIN
1274  12   15:3    13       INUM:=TOTAL5;
1275  12   15:3    17       WRITE(LISTFILE,INUM:8,INUM/OPTOTAL*100:16:2,' % ');
1276  12   15:3    60       IF TOTAL5<>0 THEN
1277  12   15:4    66         BEGIN
1278  12   15:5    66           HISTOGRAM(TOTAL5);
1279  12   15:5    70           WRITELN(LISTFILE,CR); WRITEHDR(LISTFILE,5);
1280  12   15:5    94           WRITELN(LISTFILE); WRITEHDR(LISTFILE,6);
1281  12   15:5   108           FOR I:=0 TO 15 DO
1282  12   15:6   119             WRITELN(LISTFILE,I+1:5,PARMONE5[I]:13,PARMONE5[I]/TOTAL5*100:14:2,
1283  12   15:6   177               PARMTWO5[I]:9,PARMTWO5[I]/TOTAL5*100:14:2,
1284  12   15:6   223               PARMTHREE5[I]:9,PARMTHREE5[I]/TOTAL5*100:14:2);
1285  12   15:4   284         END
1286  12   15:3   284       ELSE WRITELN(LISTFILE);
1287  12   15:2   294     END;
1288  12   15:0   294 END;
1289  12   15:0   312
1290  12   16:0     1 PROCEDURE CMPRSSST;
1291  12   16:0     1 VAR   I:INTEGER;
1292  12   16:0     0 BEGIN
1293  12   16:1     0   WITH OPCODE[OP]^ DO
1294  12   16:2    13     BEGIN
1295  12   16:3    13       WRITE(LISTFILE,TOTAL6:8,TOTAL6/OPTOTAL*100:16:2,' % ');
1296  12   16:3    58       HISTOGRAM(TOTAL6);
1297  12   16:3    62       WRITELN(LISTFILE,CR); WRITEHDR(LISTFILE,7);
1298  12   16:3    96       IF TOTAL6=0 THEN
1299  12   16:4    92         BEGIN
1300  12   16:5    92           FOR I:=0 TO 19 DO
1301  12   16:6   103             WRITELN(LISTFILE,NAMES[86+I],FLAVOR6[I]:9,0.0:14:2,'      ',
1302  12   16:6   179                     NAMES[106+I],FLAVOR6[I+20]:9,0.0:14:2);
1303  12   16:5   255           WRITELN(LISTFILE,NAMES[126]:44,FLAVOR6[40]:9,0.0:14:2);
1304  12   16:4   320         END
1305  12   16:3   320       ELSE
1306  12   16:4   322         BEGIN
1307  12   16:5   322           FOR I:=0 TO 19 DO
1308  12   16:6   333             WRITELN(LISTFILE,NAMES[86+I],FLAVOR6[I]:9,
1309  12   16:6   377               FLAVOR6[I]/TOTAL6*100:14:2,
1310  12   16:6   404               NAMES[106+I]:1[illegible]FLAVOR6[I+20]:9,FLAVOR6[I+20]/TOTAL6*100[illegible]:2);
```

```
1311  12   16:5   494            WRITELN(LISTFILE,NAMES[126]:44,
1312  12   16:5   517                FLAVOR6[40]:9,FLAVOR6[40]/TOTAL6*100:14:2);
1313  12   16:4   571          END;
1314  12   16:2   571      END;
1315  12   16:0   571 END;
1316  12   16:0   596
1317  12   17:0     1 PROCEDURE CMPRSS2ST;
1318  12   17:0     1 VAR   I:INTEGER;
1319  12   17:0     0 BEGIN
1320  12   17:1     0   WITH OPCODE[OP]^ DO
1321  12   17:2    13   BEGIN
1322  12   17:3    13     INUM:=TOTAL7;
1323  12   17:3    17     WRITE(LISTFILE,INUM:8,INUM/OPTOTAL*100:16:2,' % ');
1324  12   17:3    60     HISTOGRAM(TOTAL7);
1325  12   17:3    64     WRITELN(LISTFILE,CR); WRITEHDR(LISTFILE,7);
1326  12   17:3    88     FOR I:=1 TO 6 DO
1327  12   17:4    99       BEGIN
1328  12   17:5    99         IF INUM<>0 THEN
1329  12   17:6   104           WRITE(LISTFILE,NAMES[51+I],FLAVOR7[I]:9,FLAVOR7[I]/INUM*100:14:2,'     ')
1330  12   17:5   195         ELSE
1331  12   17:6   197           WRITE(LISTFILE,NAMES[51+I],FLAVOR7[I]:9,0.0:14:2,'     ');
1332  12   17:5   275         IF (I MOD 2=0) THEN WRITELN(LISTFILE);
1333  12   17:4   290       END;
1334  12   17:2   297   END;
1335  12   17:0   297 END;
1336  12   17:0   314
1337  12   18:0     1 PROCEDURE GINIT;
1338  12   18:0     0 BEGIN
1339  12   18:1     0   MAXOP:=0;
1340  12   18:1     3   FOR OP:=128 TO 215 DO
1341  12   18:2    18     WITH OPCODE[OP]^ DO
1342  12   18:3    31       CASE RECTYPES[OP] OF
1343  12   18:3    46             ONE,CHRS,BLK:IF (TOTAL1>MAXOP) THEN MAXOP:=TOTAL1;
1344  12   18:3    58                      TWO:IF (TOTAL2>MAXOP) THEN MAXOP:=TOTAL2;
1345  12   18:3    70                 WORD,OPT:IF (TOTAL3>MAXOP) THEN MAXOP:=TOTAL3;
1346  12   18:3    82                     LOPT:IF (TOTAL4>MAXOP) THEN MAXOP:=TOTAL4;
1347  12   18:3    94                    WORDS:IF (TOTAL5>MAXOP) THEN MAXOP:=TOTAL5;
1348  12   18:3   106                   CMPRSS:IF (TOTAL6>MAXOP) THEN MAXOP:=TOTAL6;
1349  12   18:3   118                  CMPRSS2:IF (TOTAL7>MAXOP) THEN MAXOP:=TOTAL7
1350  12   18:3   124        END;
1351  12   18:0   165 END;
```

```
1352  12   18:0   182
1353  12    1:0     0 BEGIN (* SEGMENT PROCEDURE GATHER *)
1354  12    1:1     0   GINIT;
1355  12    1:1     2   PAGE(OUTPUT);
1356  12    1:1    12   GOTOXY(0,10);
1357  12    1:1    17   WRITE(CHR(7),'OUTPUT FILE FOR OPCODE STATISTICS (<CR> FOR NONE): ');
1358  12    1:1    90   READLN(FILENAME);
1359  12    1:1   109   DISPLAY:=(FILENAME<>'');
1360  12    1:1   118   CONSOLE:=(FILENAME='CONSOLE:') OR (FILENAME='#1:');
1361  12    1:1   146   IF DISPLAY THEN
1362  12    1:2   149     BEGIN
1363  12    1:3   149       IF (FILENAME<>LASTFILENAME) THEN
1364  12    1:4   158         BEGIN
1365  12    1:5   158           CLOSE(LISTFILE,LOCK);
1366  12    1:5   167           REWRITE(LISTFILE,FILENAME);
1367  12    1:5   179           LASTFILENAME:=FILENAME;
1368  12    1:4   186         END;
1369  12    1:3   186       PAGE(OUTPUT);
1370  12    1:3   196       PROCSTUFF;
1371  12    1:3   198       JUMPSTUFF;
1372  12    1:3   200       SHORTSTUFF;
1373  12    1:3   202       FOR OP:=128 TO 215 DO
1374  12    1:4   218         BEGIN
1375  12    1:5   218           WRITE(LISTFILE,CR,NAMES[OP],'  OPCODE:',OP:4,'   TOTAL:');
1376  12    1:5   303           CASE RECTYPES[OP] OF
1377  12    1:5   318                       SHORT:SHORTST;
1378  12    1:5   322                   OPT,WORD:WORDST;
1379  12    1:5   326                ONE,CHRS,BLK:ONEST;
1380  12    1:5   330                         TWO:TWOST;
1381  12    1:5   334                        LOPT:LOPTST;
1382  12    1:5   338                       WORDS:WORDSST;
1383  12    1:5   342                      CMPRSS:CMPRSSST;
1384  12    1:5   346                     CMPRSS2:CMPRSS2ST
1385  12    1:5   346             END;
1386  12    1:4   380         END;
1387  12    1:3   387       WRITELN(LISTFILE,CR,CR,CR,OPTOTAL:20,'   TOTAL OPERATORS');
1388  12    1:3   466       WRITELN(CR,CR,CR,OPTOTAL:20,'   TOTAL OPERATORS');
1389  12    1:2   545     END;
1390  12    1:0   545 END;
1391  12    1:0   564
1392  12    1:D     1 SEGMENT PROCEDURE DATACOUNT
```

```
1393  13    1:D     1 TYPE  ACTPTR=^ACTREC;
1394  13    1:D     1       ACTREC=RECORD
1395  13    1:D     1         OFFSET,TOTAL:INTEGER;
1396  13    1:D     1         LES,GTR:ACTPTR
1397  13    1:D     1       END;
1398  13    1:D     1 VAR   TOTAL:INTEGER;
1399  13    1:D     2       HEAP:^INTEGER;
1400  13    1:D     3       TREETRUNK,ENTRY:ACTPTR;
1401  13    1:D     5       FILENAME:STRING;
1402  13    1:D    46
1403  13    2:D     1 PROCEDURE SETORDER;
1404  13    2:D     1 VAR INDEX:INTEGER;
1405  13    2:D     2
1406  13    3:D     1 PROCEDURE DATASET(TREEMARK:ACTPTR);
1407  13    3:0     0 BEGIN
1408  13    3:0     0   [$R-]
1409  13    3:1     0   IF DSSTART^[INDEX]<TREEMARK^.TOTAL THEN
1410  13    3:2    12     IF TREEMARK^.LES<>NIL THEN
1411  13    3:3    18       DATASET(TREEMARK^.LES)
1412  13    3:2    20     ELSE
1413  13    3:3    24       BEGIN
1414  13    3:4    24         NEW(ENTRY);
1415  13    3:4    30         ENTRY^.OFFSET:=INDEX;
1416  13    3:4    39         ENTRY^.TOTAL:=DSSTART^[INDEX];
1417  13    3:4    50         ENTRY^.LES:=NIL;
1418  13    3:4    57         ENTRY^.GTR:=NIL;
1419  13    3:4    64         TREEMARK^.LES:=ENTRY;
1420  13    3:3    71       END
1421  13    3:1    71   ELSE IF TREEMARK^.GTR<>NIL THEN
1422  13    3:3    79       DATASET(TREEMARK^.GTR)
1423  13    3:2    81     ELSE
1424  13    3:3    85       BEGIN
1425  13    3:4    85         NEW(ENTRY);
1426  13    3:4    91         ENTRY^.OFFSET:=INDEX;
1427  13    3:4   100         ENTRY^.TOTAL:=DSSTART^[INDEX];
1428  13    3:4   111         ENTRY^.LES:=NIL;
1429  13    3:4   118         ENTRY^.GTR:=NIL;
1430  13    3:4   125         TREEMARK^.GTR:=ENTRY;
1431  13    3:3   132       END;
1432  13    3:3   132   [$R+]
1433  13    3:0   132 END;
```

```
1434  13    3:0   144
1435  13    2:0     0 BEGIN
1436  13    2:1     0   NEW(TREETRUNK);
1437  13    2:1     6   TREETRUNK^.TOTAL:=0;
1438  13    2:1    11   TREETRUNK^.LES:=NIL;
1439  13    2:1    18   TREETRUNK^.GTR:=NIL;
1440  13    2:1    25   DATAREF:=0; INDEX:=0;
1441  13    2:1    31   REPEAT
1442  13    2:1    31     [$R-]
1443  13    2:2    31     INDEX:=INDEX + SCAN((DATASEGSIZE-INDEX)*2,<>CHR(0),DSSTART^[INDEX]) DIV 2;
1444  13    2:2    53     IF DSSTART^[INDEX]>0 THEN
1445  13    2:3    62       BEGIN
1446  13    2:4    62         DATASET(TREETRUNK);
1447  13    2:4    67         DATAREF:=DATAREF + DSSTART^[INDEX];
1448  13    2:4    77         DSSTART^[INDEX]:=0;
1449  13    2:3    83       END;
1450  13    2:3    83     [$R+]
1451  13    2:1    83   UNTIL INDEX>=DATASEGSIZE;
1452  13    2:0    89 END;
1453  13    2:0   104
1454  13    4:D     1 PROCEDURE DATAHEADER(VAR H2:INTERACTIVE);
1455  13    4:D     2 VAR  I:INTEGER;
1456  13    4:0     0 BEGIN
1457  13    4:1     0   WRITELN(H2,CR,CR,'DATA SEGMENT SIZE:',DATASEGSIZE:6,'      DATA REFERENCES:',
1458  13    4:1    85                          DATAREF:6,'      LEX LEVEL',PROCLEX[DATAPROC]:6);
1459  13    4:1   143   WRITE(H2,CR,CR,'FOR SEGMENT ');
1460  13    4:1   181   FOR I:=1 TO 8 DO WRITE(H2,CHR(SEGDIREC[63 + DATASEG*8 +I]));
1461  13    4:1   223   WRITELN(H2,' PROCEDURE #',DATAPROC:3);
1462  13    4:1   260   WRITELN(H2,'OFFSET(WORD)  TOTAL      %');
1463  13    4:0   302 END;
1464  13    4:0   316
1465  13    5:D     1 PROCEDURE PRINTDATA(TREE:ACTPTR);
1466  13    5:0     0 BEGIN
1467  13    5:1     0   IF TREE^.GTR<>NIL THEN PRINTDATA(TREE^.GTR);
1468  13    5:1    10   TOTAL:=TREE^.TOTAL;
1469  13    5:1    15   IF DISPLAY THEN WRITELN(LISTFILE,
1470  13    5:2    18              TREE^.OFFSET:9,TOTAL:11,TOTAL/DATAREF*100:9:2);
1471  13    5:1    69   IF TREE^.LES<>NIL THEN PRINTDATA(TREE^.LES);
1472  13    5:0    79 END;
1473  13    5:0    92
1474  13    1:0     0 BEGIN (* DATACOUNT *);
```

```
1475  13    1:1     0   MARK(HEAP);
1476  13    1:1     4   PAGE(OUTPUT);
1477  13    1:1    14   GOTOXY(0,10);
1478  13    1:1    19   WRITE(CHR(7),'OUTPUT FILE FOR DATA SEGMENT STATISTICS(<CR> FOR NONE): ');
1479  13    1:1    97   READLN(FILENAME);
1480  13    1:1   116   DISPLAY:=(FILENAME<>'');
1481  13    1:1   125   CONSOLE:=(FILENAME='CONSOLE:') OR (FILENAME='#1:');
1482  13    1:1   153   IF DISPLAY AND (FILENAME<>LASTFILENAME) THEN
1483  13    1:2   164     BEGIN
1484  13    1:3   164       CLOSE(LISTFILE,LOCK);
1485  13    1:3   173       REWRITE(LISTFILE,FILENAME);
1486  13    1:3   185       LASTFILENAME:=FILENAME;
1487  13    1:2   192     END;
1488  13    1:1   192   PAGE(OUTPUT);
1489  13    1:1   202   SETORDER;
1490  13    1:1   204   IF DISPLAY THEN DATAHEADER(LISTFILE);
1491  13    1:1   212   IF DATAREF>0 THEN
1492  13    1:2   218     PRINTDATA(TREETRUNK^.GTR)
1493  13    1:1   220   ELSE
1494  13    1:2   224     BEGIN
1495  13    1:3   224       IF DISPLAY THEN WRITELN(LISTFILE,CR,CR,
1496  13    1:4   247                 'SORRY BUT THERE WERE NO ACCESSES',
1497  13    1:4   291                 ' TO THIS DATA SEGMENT FROM DIS-ASSEMBLED PROCEDURES');
1498  13    1:2   362     END;
1499  13    1:1   362   PROMPT;
1500  13    1:1   365   RELEASE(HEAP);
1501  13    1:0   369 END;
1502  13    1:0   386
1503   1    2:0     1 PROCEDURE PROMPT;
1504   1    2:0     1 VAR  CH:CHAR;
1505   1    2:0     0 BEGIN
1506   1    2:1     0   WRITE(CHR(7),CR,CR,'PRESS SPACEBAR TO CONTINUE...');
1507   1    2:1    71   REPEAT  READ(CH)  UNTIL CH=' ';
1508   1    2:1    86   WRITELN;
1509   1    2:0    94 END;
1510   1    2:0   108
1511   1    2:0   108 (*$I DISASM2.TEXT*)
1512   1    2:0   118
1513   1    1:0     0 BEGIN(*MAIN STUFF*)
1514   1    1:1     0   INIT;
1515   1    1:1    29   DISASSEMBLE;
```

```
1516    1     1:1     32   IF DATAWATCH THEN DATACOUNT;
1517    1     1:1     39   GATHER;
1518    1     1:1     42   IF (LASTFILENAME<>'') AND NOT CONSOLE THEN CLOSE(LISTFILE,LOCK);
1519    1     1:0     65 END.
```