```
0001:  (*$U-*)
0002:  PROGRAM PASCALSYSTEM;  (* VERSION I.4A  2-26-78 *)
0003:
0004:
0005:  (*********************************************)
0006:  (*                                           *)
0007:  (*        UCSD  PASCAL  COMPILER             *)
0008:  (*                                           *)
0009:  (*    BASED ON ZURICH P2 PORTABLE            *)
0010:  (*    COMPILER, EXTENSIVELY                  *)
0011:  (*    MODIFIED BY ROGER T. SUMNER            *)
0012:  (*    1976..1977                             *)
0013:  (*                                           *)
0014:  (*    RELEASE LEVEL: I.3 AUGUST, 1977        *)
0015:  (*                   I.4 JANUARY, 1978       *)
0016:  (*                                           *)
0017:  (*    INSTITUTE FOR INFORMATION SYSTEMS      *)
0018:  (*    UC SAN DIEGO, LA JOLLA, CA             *)
0019:  (*                                           *)
0020:  (*    KENNETH L. BOWLES, DIRECTOR            *)
0021:  (*                                           *)
0022:  (*    COPYRIGHT (C) 1978, REGENTS OF THE     *)
0023:  (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO    *)
0024:  (*                                           *)
0025:  (*********************************************)
0026:
0027:  TYPE PHYLE = FILE;
0028:       INFOREC = RECORD
0029:                   WORKSYM,WORKCODE: ^PHYLE;
0030:                   ERRSYM,ERRBLK,ERRNUM: INTEGER;
0031:                   SLOWTERM,STUPID: BOOLEAN;
0032:                   ALTMODE: CHAR
0033:                 END;
0034:
0035:
0036:
0037:  SEGMENT PROCEDURE USERPROGRAM; BEGIN END;
0038:
0039:  SEGMENT PROCEDURE FILEHANDLER; BEGIN END;
0040:
0041:  SEGMENT PROCEDURE DEBUGGER;    BEGIN END;
0042:
0043:  SEGMENT PROCEDURE PRINTERROR;  BEGIN END;
0044:
0045:  SEGMENT PROCEDURE INITIALIZE;  BEGIN END;
0046:
0047:  SEGMENT PROCEDURE GETCMD;      BEGIN END;
0048:
0049:  SEGMENT PROCEDURE SEGMENT7;    BEGIN END;
0050:
0051:  SEGMENT PROCEDURE SEGMENT8;    BEGIN END;
0052:
0053:  SEGMENT PROCEDURE SEGMENT9;    BEGIN END;
0054:
0055:
0056:  SEGMENT PROCEDURE PASCALCOMPILER(VAR USERINFO: INFOREC);(* <<<< SMF 2-25-78 *)
0057:
0058:  CONST DISPLIMIT = 12; MAXLEVEL = 8; MAXADDR = 28000;
0059:        INTSIZE = 1; REALSIZE = 2; BITSPERWD = 16;
0060:        CHARSIZE = 1; BOOLSIZE = 1; PTRSIZE = 1;
0061:        FILESIZE = 300; NILFILESIZE = 40; BITSPERCHR = 8; CHRSPERWD = 2;
0062:        STRINGSIZE = 0; STRGLGTH = 255; MAXINT = 32767;
0063:        DEFSTRGLGTH = 80; LCAFTERMARKSTACK = 1;
0064:        EOL = 13; MAXCURSOR = 1023; MAXCODE = 1299;
0065:        MAXJTAB = 24; MAXSEG = 15; MAXPROCNUM = 149;
0066:
```

```
0067:  TYPE
0068:                                                 (*BASIC SYMBOLS*)
0069:
0070:       SYMBOL =  (IDENT,COMMA,COLON,SEMICOLON,LPARENT,RPARENT,DOSY,TOSY,
0071:                  DOWNTOSY,ENDSY,UNTILSY,OFSY,THENSY,ELSESY,BECOMES,LBRACK,
0072:                  RBRACK,ARROW,PERIOD,BEGINSY,IFSY,CASESY,REPEATSY,WHILESY,
0073:                  FORSY,WITHSY,GOTOSY,LABELSY,CONSTSY,TYPESY,VARSY,PROCSY,
0074:                  FUNCSY,PROGSY,FORWARDSY,INTCONST,REALCONST,STRINGCONST,
0075:                  NOTSY,MULOP,ADDOP,RELOP,SETSY,PACKEDSY,ARRAYSY,RECORDSY,
0076:                  FILESY,OTHERSY);
0077:
0078:
0079:       OPERATOR = (MUL,RDIV,ANDOP,IDIV,IMOD,PLUS,MINUS,OROP,LTOP,LEOP,
0080:                   GEOP,GTOP,NEOP,EQOP,INOP,NOOP);
0081:
0082:       SETOFSYS = SET OF SYMBOL;
0083:
0084:                                                 (*CONSTANTS*)
0085:       CSTCLASS = (REEL,PSET,STRG,TRIX);
0086:       CSP = ^ CONSTREC;
0087:       CONSTREC = RECORD CASE CCLASS: CSTCLASS OF
0088:                           TRIX: (CSTVAL: ARRAY [1..8] OF INTEGER);
0089:                           REEL: (RVAL: REAL);
0090:                           PSET: (PVAL: SET OF 0..127);
0091:                           STRG: (SLGTH: 0..STRGLGTH;
0092:                                  SVAL: PACKED ARRAY [1..STRGLGTH] OF CHAR)
0093:                         END;
0094:
0095:       VALU = RECORD CASE BOOLEAN OF
0096:                       TRUE:  (IVAL: INTEGER);
0097:                       FALSE: (VALP: CSP)
0098:                     END;
0099:
0100:                                                   (*DATA STRUCTURES*)
0101:       BITRANGE = 0..BITSPERWD; OPRANGE = 0..127;
0102:       CURSRANGE = 0..MAXCURSOR; PROCRANGE = 0..MAXPROCNUM;
0103:       LEVRANGE = 0..MAXLEVEL; ADDRRANGE = 0..MAXADDR;
0104:       JTABRANGE = 0..MAXJTAB; SEGRANGE = 0..MAXSEG;
0105:       DISPRANGE = 0..DISPLIMIT;
0106:
0107:       STRUCTFORM = (SCALAR,SUBRANGE,POINTER,POWER,ARRAYS,
0108:                     RECORDS,FILES,TAGFLD,VARIANT);
0109:
0110:       DECLKIND = (STANDARD,DECLARED,SPECIAL);
0111:
0112:       STP = ^ STRUCTURE; CTP = ^ IDENTIFIER;
0113:
0114:       STRUCTURE = RECORD
0115:                     SIZE: ADDRRANGE;
0116:                     CASE FORM: STRUCTFORM OF
0117:                       SCALAR:   (CASE SCALKIND: DECLKIND OF
0118:                                    DECLARED: (FCONST: CTP));
0119:                       SUBRANGE: (RANGETYPE: STP; MIN,MAX: VALU);
0120:                       POINTER:  (ELTYPE: STP);
0121:                       POWER:    (ELSET: STP);
0122:                       ARRAYS:   (AELTYPE,INXTYPE: STP;
0123:                                  CASE AISPACKD:BOOLEAN OF
0124:                                    TRUE: (ELSPERWD,ELWIDTH: BITRANGE;
0125:                                           CASE AISSTRNG: BOOLEAN OF
0126:                                            TRUE:(MAXLENG: 1..STRGLGTH)));
0127:                       RECORDS:  (FSTFLD: CTP; RECVAR: STP);
0128:                       FILES:    (FILTYPE: STP);
0129:                       TAGFLD:   (TAGFIELDP: CTP; FSTVAR: STP);
0130:                       VARIANT:  (NXTVAR,SUBVAR: STP; VARVAL: VALU)
0131:                     END;
0132:
```

```
0133:                                                                (*NAMES*)
0134:       IDCLASS = (TYPES,KONST,VARS,FIELD,PROC,FUNC);
0135:       SETOFIDS = SET OF IDCLASS;
0136:       IDKIND = (ACTUAL,FORMAL);
0137:       ALPHA = PACKED ARRAY [1..8] OF CHAR;
0138:
0139:       IDENTIFIER = RECORD
0140:                      NAME: ALPHA; LLINK, RLINK: CTP;
0141:                      IDTYPE: STP; NEXT: CTP;
0142:                      CASE KLASS: IDCLASS OF
0143:                        KONST: (VALUES: VALU);
0144:                        VARS:  (VKIND: IDKIND; VLEV: LEVRANGE;
0145:                                VADDR: ADDRRANGE);
0146:                        FIELD: (FLDADDR: ADDRRANGE;
0147:                                CASE FISPACKD: BOOLEAN OF
0148:                                  TRUE: (FLDRBIT,FLDWIDTH: BITRANGE));
0149:                        PROC,
0150:                        FUNC:  (CASE PFDECKIND: DECLKIND OF
0151:                                  SPECIAL:  (KEY: INTEGER);
0152:                                  STANDARD: (CSPNUM: INTEGER);
0153:                                  DECLARED: (PFLEV: LEVRANGE;
0154:                                             PFNAME: PROCRANGE;
0155:                                             PFSEG: SEGRANGE;
0156:                                             CASE PFKIND: IDKIND OF
0157:                                               ACTUAL: (LOCALLC: ADDRRANGE;
0158:                                                        FORWDECL,
0159:                                                        INSCOPE: BOOLEAN)))
0160:                      END;
0161:
0162:
0163:       WHERE = (BLCK,CREC,VREC,REC);
0164:
0165:                                              (*EXPRESSIONS*)
0166:       ATTRKIND = (CST,VARBL,EXPR);
0167:       VACCESS = (DRCT,INDRCT,PACKD,MULTI,BYTE);
0168:
0169:       ATTR = RECORD TYPTR: STP;
0170:                CASE KIND: ATTRKIND OF
0171:                  CST:   (CVAL: VALU);
0172:                  VARBL: (CASE ACCESS: VACCESS OF
0173:                            DRCT:   (VLEVEL: LEVRANGE; DPLMT: ADDRRANGE);
0174:                            INDRCT: (IDPLMT: ADDRRANGE))
0175:              END;
0176:
0177:       TESTP = ^ TESTPOINTER;
0178:       TESTPOINTER = RECORD
0179:                       ELT1,ELT2 : STP;
0180:                       LASTTESTP : TESTP
0181:                     END;
0182:
0183:                                                    (*LABELS*)
0184:       LBP = ^ CODELABEL;
0185:       CODELABEL = RECORD
0186:                     CASE DEFINED: BOOLEAN OF
0187:                       FALSE: (REFLIST: ADDRRANGE);
0188:                       TRUE:  (OCCURIC: ADDRRANGE; JTABINX: JTABRANGE)
0189:                   END;
0190:
0191:       LABELP = ^ USERLABEL;
0192:       USERLABEL = RECORD
0193:                     LABVAL: INTEGER;
0194:                     NEXTLAB: LABELP;
0195:                     CODELBP: LBP
0196:                   END;
0197:
0198:       CODEARRAY = PACKED ARRAY [0..MAXCODE] OF CHAR;
```

```
0199:       SYMBUFARRAY = PACKED ARRAY [CURSRANGE] OF CHAR;
0200:
0201:  (*-------------------------------------------------------------------*)
0202:
0203:  VAR
0204:
0205:      CODEP: ^ CODEARRAY;             (*CODE BUFFER UNTIL WRITEOUT*)
0206:      SYMBUFP: ^ SYMBUFARRAY;         (*SYMBOLIC BUFFER...ASCII OR CODED*)
0207:
0208:      GATTR: ATTR;                    (*DESCRIBES CURRENT EXPRESSION*)
0209:      VAL: VALU;                      (*VALUE OF LAST CONSTANT*)
0210:
0211:      DISX,                           (*LEVEL OF LAST ID SEARCHED*)
0212:      TOP: DISPRANGE;                 (*TOP OF DISPLAY*)
0213:                                      (*SCANNER GLOBALS...NEXT FOUR VARS*)
0214:                                      (*MUST BE IN THIS ORDER FOR IDSEARCH*)
0215:      SYMCURSOR: CURSRANGE;           (*CURRENT SCANNING INDEX IN SYMBUFP^*)
0216:      SY: SYMBOL;                     (*SYMBOL FOUND BY INSYMBOL*)
0217:      OP: OPERATOR;                   (*CLASSIFICATION OF LAST SYMBOL*)
0218:      ID: ALPHA;                      (*LAST IDENTIFIER FOUND*)
0219:
0220:      LGTH: INTEGER;                  (*LENGTH OF LAST STRING CONSTANT*)
0221:
0222:      LCMAX,LC,IC: ADDRRANGE;         (*LOCATION AND INSTRUCT COUNTERS*)
0223:
0224:                                      (*SWITCHES:*)
0225:
0226:      PRTERR,GOTOOK,RANGECHECK,DEBUGGING,
0227:      NOISY,CODEINSEG,IOCHECK,BPTONLINE,
0228:      LIST,TEST,SYSCOMP,DP,INCLUDING: BOOLEAN;
0229:      PMDLEVEL: INTEGER;
0230:
0231:                                      (*POINTERS:*)
0232:      INTPTR,REALPTR,
0233:      CHARPTR,BOOLPTR,
0234:      TEXTPTR,NILPTR,
0235:      INTRACTVPTR,STRGPTR: STP;       (*POINTERS TO STANDARD IDS*)
0236:
0237:      UTYPPTR,UCSTPTR,UVARPTR,
0238:      UFLDPTR,UPRCPTR,UFCTPTR,        (*POINTERS TO UNDECLARED IDS*)
0239:      INPUTPTR,OUTPUTPTR,
0240:      OUTERBLOCK,FWPTR: CTP;
0241:
0242:      GLOBTESTP: TESTP;                (*LAST TESTPOINTER*)
0243:
0244:      LEVEL: LEVRANGE;                (*CURRENT STATIC LEVEL*)
0245:
0246:      SEG,NEXTSEG: SEGRANGE;          (*CURRENT SEGMENT #*)
0247:      SEGINX: INTEGER;                (*CURRENT INDEX IN SEGMENT*)
0248:      SCONST: CSP;                    (*INSYMBOL STRING RESULTS*)
0249:
0250:      LOWTIME,LINEINFO,SCREENDOTS,STARTDOTS,SYMBLK: INTEGER;
0251:      LINESTART: CURSRANGE;
0252:
0253:      CURPROC,NEXTPROC: PROCRANGE;     (*PROCEDURE NUMBER ASSIGNMENT*)
0254:
0255:      CONSTBEGSYS,SIMPTYPEBEGSYS,TYPEBEGSYS,BLOCKBEGSYS,
0256:      SELECTSYS,FACBEGSYS,STATBEGSYS,TYPEDELS: SETOFSYS;
0257:
0258:      DISPLAY: ARRAY [DISPRANGE] OF
0259:                  RECORD
0260:                    FNAME: CTP;
0261:                    CASE OCCUR: WHERE OF
0262:                      BLCK: (FFILE: CTP; FLABEL: LABELP);
0263:                      CREC: (CLEV: LEVRANGE; CDSPL: ADDRRANGE);
0264:                      VREC: (VDSPL: ADDRRANGE)
```

```
0265:                   END;
0266:
0267:      PROCTABLE: ARRAY [PROCRANGE] OF INTEGER;
0268:
0269:      SEGTABLE: ARRAY [SEGRANGE] OF
0270:                  RECORD
0271:                    DISKADDR,CODELENG: INTEGER;
0272:                    SEGNAME: ALPHA
0273:                  END (*SEGTABLE*) ;
0274:
0275:      NEXTJTAB: JTABRANGE;
0276:      JTAB: ARRAY [JTABRANGE] OF INTEGER;
0277:
0278:      OLDSYMBLK: INTEGER;
0279:      OLDSYMCURSOR,OLDLINESTART: CURSRANGE;
0280:      INCLFILE: FILE;
0281:      LP: TEXT;
0282:
0283:      CURBYTE, CURBLK: INTEGER;
0284:      DISKBUF: PACKED ARRAY [0..511] OF CHAR;
0285:
0286:  (*-------------------------------------------------------------------*)
0287:
0288:  SEGMENT PROCEDURE COMPINIT;
0289:
0290:   (*    COPYRIGHT (C) 1978, REGENTS OF THE       *)
0291:  (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
0292:
0293:  PROCEDURE ERROR(ERRORNUM: INTEGER);
0294:    VAR CH: CHAR; ERRSTART: INTEGER;
0295:        A: PACKED ARRAY [0..179] OF CHAR;
0296:  BEGIN
0297:    WITH USERINFO DO
0298:      IF (ERRSYM <> SYMCURSOR) OR (ERRBLK <> SYMBLK) THEN
0299:        BEGIN ERRBLK := SYMBLK;
0300:          ERRSYM := SYMCURSOR; ERRNUM := ERRORNUM;
0301:          IF NOISY THEN WRITELN(OUTPUT)
0302:          ELSE
0303:            IF LIST AND (ERRORNUM <= 400) THEN
0304:              EXIT(ERROR);
0305:          IF LINESTART = 0 THEN
0306:            WRITE(OUTPUT,SYMBUFP^:SYMCURSOR)
0307:          ELSE
0308:            BEGIN
0309:              ERRSTART := SCAN(-(LINESTART-1),=CHR(EOL),
0310:                               SYMBUFP^[LINESTART-2])+LINESTART-1;
0311:              MOVELEFT(SYMBUFP^[ERRSTART],A[0],SYMCURSOR-ERRSTART);
0312:              WRITE(OUTPUT,A:SYMCURSOR-ERRSTART)
0313:            END;
0314:          WRITELN(OUTPUT,' <<<<');
0315:          WRITE(OUTPUT,'LINE ',SCREENDOTS,', ERROR ',ERRORNUM:0,':');
0316:          IF NOISY THEN
0317:            WRITE(OUTPUT,' <SP>(CONTINUE), <ESC>(TERMINATE), E(DIT');
0318:          REPEAT READ(KEYBOARD,CH)
0319:          UNTIL (CH = ' ') OR (CH = 'E') OR (CH = ALTMODE);
0320:          IF CH = 'E' THEN
0321:            BEGIN ERRBLK := SYMBLK-2; EXIT(PASCALCOMPILER) END; (* <<<< SMF *)
0322:          IF (ERRORNUM > 400) OR (CH = CHR(27)) THEN
0323:            BEGIN ERRBLK := 0; EXIT(PASCALCOMPILER) END; (* <<<< SMF 2-25-78 *)
0324:          WRITELN(OUTPUT);
0325:          IF NOISY THEN
0326:            WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
0327:        END
0328:  END (*ERROR*) ;
0329:
0330:  PROCEDURE GETNEXTPAGE;
```

```
0331:  BEGIN SYMCURSOR := 0; LINESTART := 0;
0332:    IF INCLUDING THEN
0333:      IF BLOCKREAD(INCLFILE,SYMBUFP^,2,SYMBLK) <> 2 THEN
0334:        BEGIN CLOSE(INCLFILE); INCLUDING := FALSE;
0335:          SYMBLK := OLDSYMBLK; SYMCURSOR := OLDSYMCURSOR;
0336:          LINESTART := OLDLINESTART
0337:        END;
0338:    IF NOT INCLUDING THEN
0339:      IF BLOCKREAD(USERINFO.WORKSYM^,SYMBUFP^,2,SYMBLK) <> 2 THEN
0340:        ERROR(401);
0341:    IF SYMCURSOR = 0 THEN
0342:      IF SYMBUFP^[0] = CHR(16(*DLE*)) THEN
0343:        SYMCURSOR := 2;
0344:    SYMBLK := SYMBLK+2
0345:  END (*GETNEXTPAGE*) ;
0346:
0347:  PROCEDURE PRINTLINE;
0348:    VAR DORC,STARORC: CHAR; LENG: INTEGER;
0349:        A: PACKED ARRAY [0..99] OF CHAR;
0350:  BEGIN DORC := 'C'; STARORC := ':';
0351:    IF DP THEN DORC := 'D';
0352:    IF BPTONLINE THEN STARORC := '*';
0353:    WRITE(LP,SCREENDOTS:6,SEG:4,CURPROC:5,
0354:              STARORC,DORC,LINEINFO:6,' ');
0355:    LENG := SYMCURSOR-LINESTART;
0356:    IF LENG > 100 THEN LENG := 100;
0357:    MOVELEFT(SYMBUFP^[LINESTART],A,LENG);
0358:    IF A[0] = CHR(16(*DLE*)) THEN
0359:      BEGIN
0360:        IF A[1] > ' ' THEN
0361:          WRITE(LP,' ':ORD(A[1])-ORD(' '));
0362:        LENG := LENG-2;
0363:        MOVELEFT(A[2],A,LENG)
0364:      END;
0365:    A[LENG-1] := CHR(EOL); (*JUST TO MAKE SURE*)
0366:    WRITE(LP,A:LENG);
0367:    WITH USERINFO DO
0368:      IF (ERRBLK = SYMBLK) AND (ERRSYM > LINESTART) THEN
0369:        WRITELN(LP,'>>>>>> ERROR # ',ERRNUM)
0370:  END (*PRINTLINE*) ;
0371:
0372:  PROCEDURE ENTERID(FCP: CTP);
0373:    VAR LCP,LCP1: CTP; I: INTEGER;
0374:  BEGIN LCP := DISPLAY[TOP].FNAME;
0375:    IF LCP = NIL THEN DISPLAY[TOP].FNAME := FCP
0376:    ELSE
0377:      BEGIN I := TREESEARCH(LCP,LCP1,FCP^.NAME);
0378:        WHILE I = 0 DO
0379:          BEGIN ERROR(101);
0380:            IF LCP1^.RLINK = NIL THEN I := 1
0381:            ELSE I := TREESEARCH(LCP1^.RLINK,LCP1,FCP^.NAME)
0382:          END;
0383:        IF I = 1 THEN LCP1^.RLINK := FCP ELSE LCP1^.LLINK := FCP
0384:      END;
0385:    FCP^.LLINK := NIL; FCP^.RLINK := NIL
0386:  END (*ENTERID*) ;
0387:
0388:  PROCEDURE INSYMBOL; (* COMPILER VERSION 3.4 06-NOV-76 *)
0389:    LABEL 1;
0390:    VAR LVP: CSP; X: INTEGER;
0391:
0392:  PROCEDURE CHECKEND;
0393:  BEGIN (* CHECKS FOR THE END OF THE PAGE *)
0394:    SCREENDOTS := SCREENDOTS+1;
0395:    SYMCURSOR := SYMCURSOR + 1;
0396:    IF NOISY THEN
```

```
0397:      BEGIN WRITE(OUTPUT,'.');
0398:        IF (SCREENDOTS-STARTDOTS) MOD 50 = 0 THEN
0399:          BEGIN WRITELN(OUTPUT);
0400:            WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
0401:          END
0402:      END;
0403:    IF LIST THEN PRINTLINE;
0404:    BPTONLINE := FALSE;
0405:    IF SYMBUFP^[SYMCURSOR]=CHR(0) THEN GETNEXTPAGE
0406:    ELSE LINESTART := SYMCURSOR;
0407:    IF SYMBUFP^[SYMCURSOR] = CHR(12(*FF*)) THEN SYMCURSOR:=SYMCURSOR+1;
0408:    IF SYMBUFP^[SYMCURSOR] = CHR(16(*DLE*)) THEN
0409:      SYMCURSOR := SYMCURSOR+2
0410:    ELSE
0411:      BEGIN
0412:        SYMCURSOR := SYMCURSOR+SCAN(80,<>CHR(9),SYMBUFP^[SYMCURSOR]);
0413:        SYMCURSOR := SYMCURSOR+SCAN(80,<>' ',SYMBUFP^[SYMCURSOR])
0414:      END;
0415:    IF DP THEN LINEINFO := LC ELSE LINEINFO := IC
0416:  END;
0417:
0418:  PROCEDURE COMMENTER(STOPPER: CHAR);
0419:    VAR CH,SW,DEL: CHAR; LTITLE: STRING[40];
0420:
0421:    PROCEDURE SCANTITLE;
0422:      VAR LENG: INTEGER;
0423:    BEGIN SYMCURSOR := SYMCURSOR+2;
0424:      LENG := SCAN(40,=STOPPER,SYMBUFP^[SYMCURSOR]);
0425:      LTITLE[0] := CHR(LENG);
0426:      MOVELEFT(SYMBUFP^[SYMCURSOR],LTITLE[1],LENG);
0427:      SYMCURSOR := SYMCURSOR+LENG+1
0428:    END (*SCANTITLE*) ;
0429:
0430:  BEGIN
0431:    SYMCURSOR := SYMCURSOR+1; (* POINT TO THE FIRST CH PAST "(*" *)
0432:    IF SYMBUFP^[SYMCURSOR]='$' THEN
0433:       IF SYMBUFP^[SYMCURSOR+1] <> STOPPER THEN
0434:          REPEAT
0435:            CH := SYMBUFP^[SYMCURSOR+1];
0436:            SW := SYMBUFP^[SYMCURSOR+2];
0437:            DEL := SYMBUFP^[SYMCURSOR+3];
0438:            IF (SW = ',') OR (SW = STOPPER) THEN
0439:              BEGIN DEL := SW; SW := '+';
0440:                SYMCURSOR := SYMCURSOR-1
0441:              END;
0442:            CASE CH OF
0443:            'D': DEBUGGING := (SW='+');
0444:            'G': GOTOOK := (SW='+');
0445:            'I': IF (SW='+') OR (SW='-') THEN IOCHECK := (SW='+')
0446:                 ELSE
0447:                   BEGIN SCANTITLE;
0448:                     IF STOPPER = '*' THEN
0449:                       SYMCURSOR := SYMCURSOR+1;
0450:                     OPENOLD(INCLFILE,LTITLE);
0451:                     IF IORESULT <> 0 THEN
0452:                       BEGIN OPENOLD(INCLFILE,CONCAT(LTITLE,'.TEXT'));
0453:                          IF IORESULT <> 0 THEN ERROR(403)
0454:                       END;
0455:                     INCLUDING := TRUE;
0456:                     OLDSYMCURSOR := SYMCURSOR;
0457:                     OLDLINESTART := LINESTART;
0458:                     OLDSYMBLK := SYMBLK-2;
0459:                     SYMBLK := 2; GETNEXTPAGE;
0460:                     INSYMBOL; EXIT(INSYMBOL)
0461:                   END;
0462:            'L': IF (SW='+') OR (SW='-') THEN
```

```
0463:                   BEGIN LIST := (SW='+');
0464:                     IF LIST THEN OPENNEW(LP,'*SYSTEM.LST.TEXT')
0465:                   END
0466:                 ELSE
0467:                   BEGIN SCANTITLE;
0468:                     OPENNEW(LP,LTITLE);
0469:                     LIST := IORESULT = 0;
0470:                     EXIT(COMMENTER)
0471:                   END;
0472:             'Q': NOISY := (SW='-');
0473:             'P': WRITE(LP,CHR(12(*FF*)));
0474:             'R': RANGECHECK := (SW='+');
0475:             'U': BEGIN SYSCOMP := (SW = '-');
0476:                    RANGECHECK := NOT SYSCOMP;
0477:                    IOCHECK := RANGECHECK;
0478:                    GOTOOK := SYSCOMP
0479:                  END
0480:           END (*CASES*);
0481:           SYMCURSOR := SYMCURSOR+3;
0482:         UNTIL DEL <> ',';
0483:   SYMCURSOR := SYMCURSOR-1; (* ADJUST *)
0484:   REPEAT
0485:     REPEAT
0486:       SYMCURSOR := SYMCURSOR+1;
0487:       WHILE SYMBUFP^[SYMCURSOR] = CHR(EOL) DO CHECKEND
0488:     UNTIL SYMBUFP^[SYMCURSOR]=STOPPER;
0489:   UNTIL (SYMBUFP^[SYMCURSOR+1]=')') OR (STOPPER='');
0490:   SYMCURSOR := SYMCURSOR+1;
0491: END (*COMMENTER*);
0492:
0493: PROCEDURE STRING;
0494: LABEL 1;
0495: VAR
0496:   T: PACKED ARRAY [1..80] OF CHAR;
0497:   TP,NBLANKS,L: INTEGER;
0498:   DUPLE: BOOLEAN;
0499:
0500: BEGIN
0501:   DUPLE := FALSE; (* INDICATES WHEN '' IS PRESENT *)
0502:   TP := 0; (* INDEX INTO TEMPORARY STRING *)
0503:   REPEAT
0504:     IF DUPLE THEN SYMCURSOR := SYMCURSOR+1;
0505:     REPEAT
0506:       SYMCURSOR := SYMCURSOR+1;
0507:       TP := TP+1;
0508:       IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
0509:         BEGIN ERROR(202); CHECKEND; GOTO 1 END;
0510:       T[TP] := SYMBUFP^[SYMCURSOR];
0511:     UNTIL SYMBUFP^[SYMCURSOR]='''';
0512:     DUPLE := TRUE;
0513:   UNTIL SYMBUFP^[SYMCURSOR+1]<>'''';
0514: 1:  TP := TP-1; (* ADJUST *)
0515:   SY := STRINGCONST; OP := NOOP;
0516:   LGTH := TP; (* GROSS *)
0517:   IF TP=1 (* SINGLE CHARACTER CONSTANT *)
0518:     THEN
0519:       VAL.IVAL := ORD(T[1])
0520:     ELSE
0521:       WITH SCONST^ DO
0522:         BEGIN
0523:           CCLASS := STRG;
0524:           SLGTH := TP;
0525:           MOVELEFT(T[1],SVAL[1],TP);
0526:           VAL.VALP := SCONST
0527:         END
0528: END(*STRING*);
```

```
0529:
0530:  PROCEDURE NUMBER;
0531:  VAR
0532:    EXPONENT,ENDI,ENDF,ENDE,SIGN,IPART,FPART,EPART,
0533:    ISUM:  INTEGER;
0534:    TIPE: (REALTIPE,INTEGERTIPE);
0535:    RSUM: REAL;
0536:    J: INTEGER;
0537:  BEGIN
0538:    (* TAKES A NUMBER AND DECIDES WHETHER IT'S REAL
0539:       OR INTEGER AND CONVERTS IT TO THE INTERNAL
0540:       FORM. *)
0541:    TIPE := INTEGERTIPE;
0542:    ENDI := 0;
0543:    ENDF := 0;
0544:    ENDE := 0;
0545:    SIGN := 1;
0546:    EPART := 9999; (* OUT OF REACH *)
0547:    IPART := SYMCURSOR; (* INTEGER PART STARTS HERE *)
0548:    REPEAT
0549:      SYMCURSOR := SYMCURSOR+1
0550:    UNTIL (SYMBUFP^[SYMCURSOR]<'0') OR (SYMBUFP^[SYMCURSOR]>'9');
0551:    (* SYMCURSOR NOW POINTS AT FIRST CHARACTER PAST INTEGER PART *)
0552:    ENDI := SYMCURSOR-1; (* MARK THE END OF IPART *)
0553:    IF SYMBUFP^[SYMCURSOR]='.'
0554:      THEN
0555:        IF SYMBUFP^[SYMCURSOR+1]<>'.'  (* WATCH OUT FOR '..' *)
0556:          THEN
0557:            BEGIN
0558:              TIPE := REALTIPE;
0559:              SYMCURSOR := SYMCURSOR+1;
0560:              FPART := SYMCURSOR; (* BEGINNING OF FPART *)
0561:              WHILE (SYMBUFP^[SYMCURSOR] >= '0') AND
0562:                    (SYMBUFP^[SYMCURSOR] <= '9') DO
0563:                SYMCURSOR := SYMCURSOR+1;
0564:              IF SYMCURSOR = FPART THEN ERROR(201);
0565:              ENDF := SYMCURSOR-1;
0566:            END;
0567:    IF SYMBUFP^[SYMCURSOR]='E'
0568:      THEN
0569:        BEGIN
0570:          TIPE := REALTIPE;
0571:          SYMCURSOR := SYMCURSOR+1;
0572:          IF SYMBUFP^[SYMCURSOR]='-'
0573:            THEN
0574:              BEGIN
0575:                SYMCURSOR := SYMCURSOR+1;
0576:                SIGN := -1;
0577:              END
0578:            ELSE
0579:              IF SYMBUFP^[SYMCURSOR]='+'
0580:                THEN
0581:                  SYMCURSOR := SYMCURSOR+1;
0582:          EPART := SYMCURSOR; (* BEGINNING OF EXPONENT *)
0583:          WHILE (SYMBUFP^[SYMCURSOR]>='0') AND (SYMBUFP^[SYMCURSOR]<='9') DO
0584:            SYMCURSOR := SYMCURSOR+1;
0585:          ENDE := SYMCURSOR-1;
0586:          IF ENDE<EPART THEN ERROR(201); (* ERROR IN REAL CONSTANT *)
0587:        END;
0588:    (* NOW CONVERT TO INTERNAL FORM *)
0589:    IF TIPE=INTEGERTIPE
0590:      THEN
0591:        BEGIN
0592:          ISUM := 0;
0593:          FOR J := IPART TO ENDI DO
0594:            BEGIN
```

```
0595:              IF (ISUM>3276) OR ((ISUM=3276) AND (SYMBUFP^[J]>'7')) THEN
0596:                  BEGIN ERROR(203); J := ENDI END
0597:              ELSE ISUM := ISUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
0598:            END;
0599:            SY := INTCONST;  OP := NOOP;
0600:            VAL.IVAL := ISUM;
0601:          END
0602:        ELSE
0603:          BEGIN (* REAL NUMBER HERE *)
0604:            RSUM := 0;
0605:            FOR J := IPART TO ENDI DO
0606:              BEGIN
0607:                RSUM := RSUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
0608:              END;
0609:            FOR J := ENDF DOWNTO FPART DO
0610:              RSUM := RSUM+(ORD(SYMBUFP^[J])-ORD('0'))/PWROFTEN(J-FPART+1);
0611:            EXPONENT := 0;
0612:            FOR J := EPART TO ENDE DO
0613:              EXPONENT := EXPONENT*10+ORD(SYMBUFP^[J])-ORD('0');
0614:            IF SIGN=-1
0615:              THEN
0616:                RSUM := RSUM/PWROFTEN(EXPONENT)
0617:              ELSE
0618:                RSUM := RSUM*PWROFTEN(EXPONENT);
0619:            SY := REALCONST;  OP := NOOP;
0620:            NEW(LVP,REEL);
0621:            LVP^.CCLASS := REEL;
0622:            LVP^.RVAL := RSUM;
0623:            VAL.VALP := LVP;
0624:          END;
0625:    SYMCURSOR := SYMCURSOR-1; (* ADJUST FOR POSTERITY *)
0626:  END;
0627:
0628:  BEGIN (* INSYMBOL *)
0629:    OP := NOOP;
0630:  1:  SY := OTHERSY; (* IF NO CASES EXERCISED BLOW UP *)
0631:    CASE SYMBUFP^[SYMCURSOR] OF
0632:    '''':STRING;
0633:    '0','1','2','3','4','5','6','7','8','9':
0634:         NUMBER;
0635:    'A','B','C','D','E','F','G','H','I','J','K','L','M',
0636:    'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
0637:    'A','B','C','D','E','F','G','H','I','J','K','L','M',
0638:    'N','O','P','Q','R','S','T','U','V','W','X','Y','Z':
0639:         IDSEARCH(SYMCURSOR,SYMBUFP^); (* MAGIC PROC *)
0640:    '': BEGIN COMMENTER(''); GOTO 1 END;
0641:    '(': BEGIN
0642:           IF SYMBUFP^[SYMCURSOR+1]='*' THEN
0643:               BEGIN
0644:                  SYMCURSOR := SYMCURSOR+1;
0645:                  COMMENTER('*');
0646:                  SYMCURSOR := SYMCURSOR+1;
0647:                  GOTO 1; (* GET ANOTHER TOKEN *)
0648:               END
0649:             ELSE
0650:               SY := LPARENT;
0651:         END;
0652:    ')': SY := RPARENT;
0653:    ',': SY := COMMA;
0654:    ' ',' ': BEGIN SYMCURSOR := SYMCURSOR+1; GOTO 1; END;
0655:    '.': BEGIN
0656:           IF SYMBUFP^[SYMCURSOR+1]='.'
0657:             THEN
0658:               BEGIN
0659:                 SYMCURSOR := SYMCURSOR+1;
0660:                 SY := COLON
```

```
0661:              END
0662:            ELSE
0663:              SY := PERIOD;
0664:        END;
0665:    ':': IF SYMBUFP^[SYMCURSOR+1]='='
0666:           THEN
0667:             BEGIN
0668:               SYMCURSOR := SYMCURSOR+1;
0669:               SY := BECOMES;
0670:             END
0671:          ELSE
0672:             SY := COLON;
0673:    ';': SY := SEMICOLON;
0674:    '^': SY := ARROW;
0675:    '[': SY := LBRACK;
0676:    ']': SY := RBRACK;
0677:    '*': BEGIN SY := MULOP; OP := MUL END;
0678:    '+': BEGIN SY := ADDOP; OP := PLUS END;
0679:    '-': BEGIN SY := ADDOP; OP := MINUS END;
0680:    '/': BEGIN SY := MULOP; OP := RDIV END;
0681:    '<': BEGIN
0682:           SY := RELOP;
0683:           OP := LTOP;
0684:           CASE SYMBUFP^[SYMCURSOR+1] OF
0685:             '>': BEGIN
0686:                    OP := NEOP;
0687:                    SYMCURSOR := SYMCURSOR+1
0688:                  END;
0689:             '=': BEGIN
0690:                    OP := LEOP;
0691:                    SYMCURSOR := SYMCURSOR+1
0692:                  END
0693:           END;
0694:        END;
0695:    '=': BEGIN SY := RELOP; OP := EQOP END;
0696:    '>': BEGIN
0697:           SY := RELOP;
0698:           IF SYMBUFP^[SYMCURSOR+1]='='
0699:             THEN
0700:               BEGIN
0701:                 OP := GEOP;
0702:                 SYMCURSOR := SYMCURSOR+1;
0703:               END
0704:             ELSE
0705:               OP := GTOP;
0706:        END
0707:  END (* CASE SYMBUFP^[SYMCURSOR] OF *);
0708:    IF SY=OTHERSY THEN
0709:      IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
0710:        BEGIN CHECKEND; GOTO 1 END
0711:      ELSE ERROR(400);
0712:    SYMCURSOR := SYMCURSOR+1; (* NEXT CALL TALKS ABOUT NEXT TOKEN *)
0713:  END (*INSYMBOL*) ;
0714:
0715:
0716:    PROCEDURE ENTSTDTYPES;
0717:      VAR SP: STP;
0718:    BEGIN
0719:      NEW(INTPTR,SCALAR,STANDARD);
0720:      WITH INTPTR^ DO
0721:        BEGIN SIZE := INTSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
0722:      NEW(REALPTR,SCALAR,STANDARD);
0723:      WITH REALPTR^ DO
0724:        BEGIN SIZE := REALSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
0725:      NEW(CHARPTR,SCALAR,STANDARD);
0726:      WITH CHARPTR^ DO
```

```
0727:         BEGIN SIZE := CHARSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
0728:       NEW(BOOLPTR,SCALAR,DECLARED);
0729:       WITH BOOLPTR^ DO
0730:         BEGIN SIZE := BOOLSIZE; FORM := SCALAR; SCALKIND := DECLARED END;
0731:       NEW(NILPTR,POINTER);
0732:       WITH NILPTR^ DO
0733:         BEGIN SIZE := PTRSIZE; FORM := POINTER; ELTYPE := NIL END;
0734:       NEW(TEXTPTR,FILES);
0735:       WITH TEXTPTR^ DO
0736:         BEGIN SIZE := FILESIZE+CHARSIZE; FORM := FILES; FILTYPE := CHARPTR END;
0737:       NEW(INTRACTVPTR,FILES);
0738:       WITH INTRACTVPTR^ DO
0739:         BEGIN SIZE := FILESIZE+CHARSIZE; FORM := FILES; FILTYPE := CHARPTR END;
0740:       NEW(STRGPTR,ARRAYS,TRUE,TRUE);
0741:       WITH STRGPTR^ DO
0742:         BEGIN FORM := ARRAYS; SIZE := (DEFSTRGLGTH + CHRSPERWD) DIV CHRSPERWD;
0743:           AISPACKD := TRUE; AISSTRNG := TRUE; INXTYPE := INTPTR;
0744:           ELWIDTH := BITSPERCHR; ELSPERWD := CHRSPERWD;
0745:           AELTYPE := CHARPTR; MAXLENG := DEFSTRGLGTH;
0746:         END
0747:     END (*ENTSTDTYPES*) ;
0748:
0749:     PROCEDURE ENTSTDNAMES;
0750:       VAR CP,CP1: CTP; I: INTEGER;
0751:     BEGIN
0752:       NEW(CP,TYPES);
0753:       WITH CP^ DO
0754:         BEGIN NAME := 'INTEGER '; IDTYPE := INTPTR; KLASS := TYPES END;
0755:       ENTERID(CP);
0756:       NEW(CP,TYPES);
0757:       WITH CP^ DO
0758:         BEGIN NAME := 'REAL    '; IDTYPE := REALPTR; KLASS := TYPES END;
0759:       ENTERID(CP);
0760:       NEW(CP,TYPES);
0761:       WITH CP^ DO
0762:         BEGIN NAME := 'CHAR    '; IDTYPE := CHARPTR; KLASS := TYPES END;
0763:       ENTERID(CP);
0764:       NEW(CP,TYPES);
0765:       WITH CP^ DO
0766:         BEGIN NAME := 'BOOLEAN '; IDTYPE := BOOLPTR; KLASS := TYPES END;
0767:       ENTERID(CP);
0768:       NEW(CP,TYPES);
0769:       WITH CP^ DO
0770:         BEGIN NAME := 'STRING  '; IDTYPE := STRGPTR; KLASS := TYPES END;
0771:       ENTERID(CP);
0772:       NEW(CP,TYPES);
0773:       WITH CP^ DO
0774:         BEGIN NAME := 'TEXT    '; IDTYPE := TEXTPTR; KLASS := TYPES END;
0775:       ENTERID(CP);
0776:       NEW(CP,TYPES);
0777:       WITH CP^ DO
0778:         BEGIN NAME := 'INTERACT'; IDTYPE := INTRACTVPTR; KLASS := TYPES END;
0779:       ENTERID(CP);
0780:       NEW(CP,KONST);
0781:       WITH CP^ DO
0782:         BEGIN
0783:           NAME := 'MAXINT  '; IDTYPE := INTPTR;
0784:           KLASS := KONST; VALUES.IVAL := MAXINT
0785:         END;
0786:       ENTERID(CP);
0787:       NEW(INPUTPTR,VARS);
0788:       WITH INPUTPTR^ DO
0789:         BEGIN NAME := 'INPUT   '; IDTYPE := TEXTPTR; KLASS := VARS;
0790:           VKIND := FORMAL; VLEV := 0; VADDR := 2
0791:         END;
0792:       ENTERID(INPUTPTR);
```

```
0793:       NEW(OUTPUTPTR,VARS);
0794:       WITH OUTPUTPTR^ DO
0795:         BEGIN NAME := 'OUTPUT  '; IDTYPE := TEXTPTR; KLASS := VARS;
0796:           VKIND := FORMAL; VLEV := 0; VADDR := 3
0797:         END;
0798:       ENTERID(OUTPUTPTR);
0799:       NEW(CP,VARS);
0800:       WITH CP^ DO
0801:         BEGIN NAME := 'KEYBOARD'; IDTYPE := TEXTPTR; KLASS := VARS;
0802:           VKIND := FORMAL; VLEV := 0; VADDR := 4
0803:         END;
0804:       ENTERID(CP);
0805:       CP1 := NIL;
0806:       FOR I := 0 TO 1 DO
0807:         BEGIN NEW(CP,KONST);
0808:           WITH CP^ DO
0809:             BEGIN IDTYPE := BOOLPTR;
0810:               IF I = 0 THEN NAME := 'FALSE   '
0811:               ELSE NAME := 'TRUE    ';
0812:               NEXT := CP1; VALUES.IVAL := I; KLASS := KONST
0813:             END;
0814:           ENTERID(CP); CP1 := CP
0815:         END;
0816:       BOOLPTR^.FCONST := CP;
0817:       NEW(CP,KONST);
0818:       WITH CP^ DO
0819:         BEGIN NAME := 'NIL     '; IDTYPE := NILPTR;
0820:           NEXT := NIL; VALUES.IVAL := 0; KLASS := KONST
0821:         END;
0822:       ENTERID(CP);
0823:     END (*ENTSTDNAMES*) ;
0824:
0825:     PROCEDURE ENTUNDECL;
0826:     BEGIN
0827:       NEW(UTYPPTR,TYPES);
0828:       WITH UTYPPTR^ DO
0829:         BEGIN NAME := '        '; IDTYPE := NIL; KLASS := TYPES END;
0830:       NEW(UCSTPTR,KONST);
0831:       WITH UCSTPTR^ DO
0832:         BEGIN NAME := '        '; IDTYPE := NIL; NEXT := NIL;
0833:           VALUES.IVAL := 0; KLASS := KONST
0834:         END;
0835:       NEW(UVARPTR,VARS);
0836:       WITH UVARPTR^ DO
0837:         BEGIN NAME := '        '; IDTYPE := NIL; VKIND := ACTUAL;
0838:           NEXT := NIL; VLEV := 0; VADDR := 0; KLASS := VARS
0839:         END;
0840:       NEW(UFLDPTR,FIELD);
0841:       WITH UFLDPTR^ DO
0842:         BEGIN NAME := '        '; IDTYPE := NIL; NEXT := NIL;
0843:           FLDADDR := 0; KLASS := FIELD
0844:         END;
0845:       NEW(UPRCPTR,PROC,DECLARED,ACTUAL);
0846:       WITH UPRCPTR^ DO
0847:         BEGIN NAME := '        '; IDTYPE := NIL; FORWDECL := FALSE;
0848:           NEXT := NIL; INSCOPE := FALSE; LOCALLC := 0;
0849:           PFLEV := 0; PFNAME := 0; PFSEG := 0;
0850:           KLASS := PROC; PFDECKIND := DECLARED; PFKIND := ACTUAL
0851:         END;
0852:       NEW(UFCTPTR,FUNC,DECLARED,ACTUAL);
0853:       WITH UFCTPTR^ DO
0854:         BEGIN NAME := '        '; IDTYPE := NIL; NEXT := NIL;
0855:           FORWDECL := FALSE; INSCOPE := FALSE; LOCALLC := 0;
0856:           PFLEV := 0; PFNAME := 0; PFSEG := 0;
0857:           KLASS := FUNC; PFDECKIND := DECLARED; PFKIND := ACTUAL
0858:         END
```

```
0859:    END (*ENTUNDECL*) ;
0860:
0861:    PROCEDURE ENTSPCPROCS;
0862:      VAR LCP: CTP; I: INTEGER; ISFUNC: BOOLEAN;
0863:          NA: ARRAY [1..43] OF ALPHA;
0864:    BEGIN
0865:      NA[ 1] := 'READ    '; NA[ 2] := 'READLN  '; NA[ 3] := 'WRITE   ';
0866:      NA[ 4] := 'WRITELN '; NA[ 5] := 'EOF     '; NA[ 6] := 'EOLN    ';
0867:      NA[ 7] := 'PRED    '; NA[ 8] := 'SUCC    '; NA[ 9] := 'ORD     ';
0868:      NA[10] := 'SQR     '; NA[11] := 'ABS     '; NA[12] := 'NEW     ';
0869:      NA[13] := 'UNITREAD'; NA[14] := 'UNITWRIT'; NA[15] := 'CONCAT  ';
0870:      NA[16] := 'LENGTH  '; NA[17] := 'INSERT  '; NA[18] := 'DELETE  ';
0871:      NA[19] := 'COPY    '; NA[20] := 'POS     '; NA[21] := 'MOVELEFT';
0872:      NA[22] := 'MOVERIGH'; NA[23] := 'EXIT    '; NA[24] := 'IDSEARCH';
0873:      NA[25] := 'TREESEAR'; NA[26] := 'TIME    '; NA[27] := 'FILLCHAR';
0874:      NA[28] := 'OPENNEW '; NA[29] := 'OPENOLD '; NA[30] := 'REWRITE ';
0875:      NA[31] := 'CLOSE   '; NA[32] := 'SEEK    '; NA[33] := 'RESET   ';
0876:      NA[34] := 'GET     '; NA[35] := 'PUT     '; NA[36] := 'SCAN    ';
0877:      NA[37] := 'BLOCKREA'; NA[38] := 'BLOCKWRI'; NA[39] := 'DRAWLINE';
0878:      NA[40] := 'PAGE    '; NA[41] := 'SIZEOF  '; NA[42] := 'DRAWBLOC';
0879:      NA[43] := 'GOTOXY  ';
0880:      FOR I := 1 TO 43 DO
0881:        BEGIN ISFUNC := I IN [5,6,7,8,9,10,11,15,16,19,20,25,36,37,38,41];
0882:          IF ISFUNC THEN NEW(LCP,FUNC,SPECIAL)
0883:          ELSE NEW(LCP,PROC,SPECIAL);
0884:          WITH LCP^ DO
0885:            BEGIN NAME := NA[I]; NEXT := NIL; IDTYPE := NIL;
0886:              IF ISFUNC THEN KLASS := FUNC ELSE KLASS := PROC;
0887:              PFDECKIND := SPECIAL; KEY := I
0888:            END;
0889:          ENTERID(LCP)
0890:        END
0891:      END (*ENTSPCPROCS*) ;
0892:
0893:    PROCEDURE ENTSTDPROCS;
0894:      VAR LCP,PARAM: CTP; LSP,FTYPE: STP; I: INTEGER; ISPROC: BOOLEAN;
0895:          NA: ARRAY [1..20] OF ALPHA;
0896:    BEGIN
0897:      NA[ 1] := 'ODD     '; NA[ 2] := 'CHR     '; NA[ 3] := 'TRUNC   ';
0898:      NA[ 4] := 'ROUND   '; NA[ 5] := 'SIN     '; NA[ 6] := 'COS     ';
0899:      NA[ 7] := 'LOG     '; NA[ 8] := 'ATAN    '; NA[ 9] := 'LN      ';
0900:      NA[10] := 'EXP     '; NA[11] := 'SQRT    '; NA[12] := 'MARK    ';
0901:      NA[13] := 'RELEASE '; NA[14] := 'IORESULT'; NA[15] := 'UNITBUSY';
0902:      NA[16] := 'PWROFTEN'; NA[17] := 'UNITWAIT'; NA[18] := 'UNITCLEA';
0903:      NA[19] := 'HALT    '; NA[20] := 'MEMAVAIL';
0904:      FOR I := 1 TO 20 DO
0905:        BEGIN ISPROC := I IN [12,13,17,18,19];
0906:          CASE I OF
0907:            1:  BEGIN FTYPE := BOOLPTR; NEW(PARAM,VARS);
0908:                WITH PARAM^ DO
0909:                  BEGIN IDTYPE := INTPTR; VKIND := ACTUAL END;
0910:                END;
0911:            2:  FTYPE := CHARPTR;
0912:            3:  BEGIN FTYPE := INTPTR; NEW(PARAM,VARS);
0913:                WITH PARAM^ DO
0914:                  BEGIN IDTYPE := REALPTR; VKIND := ACTUAL END;
0915:                END;
0916:            5:  FTYPE := REALPTR;
0917:           12:  BEGIN FTYPE := NIL; NEW(PARAM,VARS); NEW(LSP,POINTER);
0918:                WITH LSP^ DO
0919:                  BEGIN SIZE := PTRSIZE; FORM := POINTER; ELTYPE := NIL END;
0920:                WITH PARAM^ DO
0921:                  BEGIN IDTYPE := LSP; VKIND := FORMAL END;
0922:                END;
0923:           14:  BEGIN FTYPE := INTPTR; PARAM := NIL END;
0924:           15:  BEGIN FTYPE := BOOLPTR; NEW(PARAM,VARS);
```

```
0925:               WITH PARAM^ DO
0926:                 BEGIN IDTYPE := INTPTR; VKIND := ACTUAL END;
0927:               END;
0928:           16:  FTYPE := REALPTR;
0929:           17:  FTYPE := NIL;
0930:           19:  BEGIN FTYPE := NIL; PARAM := NIL END;
0931:           20:  BEGIN FTYPE := INTPTR; PARAM := NIL END
0932:         END (*PARAM AND TYPE CASES*) ;
0933:         IF ISPROC THEN NEW(LCP,PROC,STANDARD)
0934:         ELSE NEW(LCP,FUNC,STANDARD);
0935:         WITH LCP^ DO
0936:           BEGIN NAME := NA[I]; PFDECKIND := STANDARD; CSPNUM := I + 20;
0937:             IF ISPROC THEN KLASS := PROC ELSE KLASS := FUNC;
0938:             IF PARAM <> NIL THEN
0939:               WITH PARAM^ DO
0940:                 BEGIN KLASS := VARS; NEXT := NIL END;
0941:             IDTYPE := FTYPE; NEXT := PARAM
0942:           END;
0943:         ENTERID(LCP)
0944:       END
0945:     END (*ENTSTDPROCS*) ;
0946:
0947:   PROCEDURE INITSCALARS;
0948:   BEGIN FWPTR := NIL; GLOBTESTP := NIL;
0949:     LINESTART := 0; LINEINFO := LCAFTERMARKSTACK; LIST := FALSE;
0950:     SYMBLK := 2; SCREENDOTS := 0; STARTDOTS := 0;
0951:     FOR SEG := 0 TO MAXSEG DO
0952:       WITH SEGTABLE[SEG] DO
0953:         BEGIN DISKADDR := 0; CODELENG := 0; SEGNAME := '        ' END;
0954:     LC := LCAFTERMARKSTACK; IOCHECK := TRUE; DP := TRUE;
0955:     SEGINX := 0; NEXTJTAB := 1; NEXTPROC := 2; CURPROC := 1;
0956:     NEW(SCONST); NEW(SYMBUFP); NEW(CODEP);
0957:     SEG := 1; NEXTSEG := 10; CURBLK := 1; CURBYTE := 0;
0958:     NOISY := NOT USERINFO.SLOWTERM;
0959:     DEBUGGING := FALSE; BPTONLINE := FALSE;
0960:     GOTOOK := FALSE; RANGECHECK := TRUE; SYSCOMP := FALSE;
0961:     CODEINSEG := FALSE; PRTERR := TRUE; INCLUDING := FALSE
0962:   END (*INITSCALARS*) ;
0963:
0964:   PROCEDURE INITSETS;
0965:   BEGIN
0966:     CONSTBEGSYS := [ADDOP,INTCONST,REALCONST,STRINGCONST,IDENT];
0967:     SIMPTYPEBEGSYS := [LPARENT] + CONSTBEGSYS;
0968:     TYPEBEGSYS := [ARROW,PACKEDSY,ARRAYSY,RECORDSY,SETSY,FILESY]
0969:                   + SIMPTYPEBEGSYS;
0970:     TYPEDELS := [ARRAYSY,RECORDSY,SETSY,FILESY];
0971:     BLOCKBEGSYS := [LABELSY,CONSTSY,TYPESY,VARSY,PROCSY,FUNCSY,PROGSY,BEGINSY];
0972:     SELECTSYS := [ARROW,PERIOD,LBRACK];
0973:     FACBEGSYS := [INTCONST,REALCONST,STRINGCONST,IDENT,LPARENT,LBRACK,NOTSY];
0974:     STATBEGSYS := [BEGINSY,GOTOSY,IFSY,WHILESY,REPEATSY,FORSY,WITHSY,CASESY]
0975:   END (*INITSETS*) ;
0976:
0977: BEGIN (*COMPINIT*)
0978:   INITSCALARS; INITSETS;
0979:   LEVEL := 0; TOP := 0;
0980:   WITH DISPLAY[0] DO
0981:     BEGIN FNAME := NIL; FFILE := NIL; FLABEL := NIL; OCCUR := BLCK END;
0982:   ENTSTDTYPES;   ENTSTDNAMES;   ENTUNDECL;
0983:   ENTSPCPROCS;   ENTSTDPROCS;
0984:   IF NOISY THEN
0985:     BEGIN
0986:       FOR IC := 1 TO 7 DO WRITELN(OUTPUT);
0987:       WRITELN(OUTPUT,'PASCAL COMPILER [I.4A]');  (* <<<< SMF 2-25-78 *)
0988:       WRITE(OUTPUT,'<   0>')
0989:     END;
0990:   GETNEXTPAGE;
```

```
0991:    INSYMBOL;
0992:    IF SYSCOMP THEN
0993:      BEGIN OUTERBLOCK := NIL; SEG := 0; NEXTSEG := 1 END
0994:    ELSE
0995:      BEGIN TOP := 1; LEVEL := 1;
0996:        WITH DISPLAY[1] DO
0997:          BEGIN FNAME := NIL; FFILE := NIL;
0998:            FLABEL := NIL; OCCUR := BLCK
0999:          END;
1000:        LC := LC+2; (*KEEP STACK STRAIGHT FOR NOW*)
1001:        NEW(OUTERBLOCK,PROC,DECLARED,ACTUAL);
1002:        WITH OUTERBLOCK^ DO
1003:          BEGIN NEXT := NIL; LOCALLC := LC;
1004:            NAME := 'PROGRAM '; IDTYPE := NIL; KLASS := PROC;
1005:            PFDECKIND := DECLARED; PFLEV := 0; PFNAME := 1; PFSEG := SEG;
1006:            PFKIND := ACTUAL; FORWDECL := FALSE; INSCOPE := TRUE
1007:          END
1008:      END;
1009:    IF SY = PROGSY THEN
1010:      BEGIN INSYMBOL;
1011:        IF SY = IDENT THEN
1012:          BEGIN SEGTABLE[SEG].SEGNAME := ID;
1013:            IF OUTERBLOCK <> NIL THEN OUTERBLOCK^.NAME := ID;
1014:            ENTERID(OUTERBLOCK);  (* SMF 2-28-78 ALLOW EXIT ON PROGRAM NAME *)
1015:          END
1016:        ELSE ERROR(2); INSYMBOL;
1017:        IF SY = LPARENT THEN
1018:          BEGIN
1019:            REPEAT INSYMBOL
1020:            UNTIL SY IN [RPARENT,SEMICOLON]+BLOCKBEGSYS;
1021:            IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
1022:          END;
1023:        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
1024:      END
1025:  END (*COMPINIT*) ;
1026:    (*      COPYRIGHT  (C) 1978, REGENTS OF THE      *)
1027:    (*      UNIVERSITY OF CALIFORNIA, SAN DIEGO      *)
1028:
1029:    SEGMENT PROCEDURE BLOCKCOMPILE;  (* <<<< SMF 2-25-78 *)
1030:    VAR DUMMYVAR:ARRAY[0..0] OF INTEGER; (*FOR PRETTY DISPLAY OF STACK AND HEAP *)
1031:
1032:    (* PROCEDURES GETNEXTPAGE, PRINTLINE, AND ENTERID
1033:       HAVE BEEN MOVED INTO BODY OF "PASCALCOMPILER" SEGMENT              *)
1034:
1035:   (*    COPYRIGHT (C) 1978, REGENTS OF THE       *)
1036:  (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
1037:
1038:  PROCEDURE ERROR(ERRORNUM: INTEGER);
1039:    VAR CH: CHAR; ERRSTART: INTEGER;
1040:        A: PACKED ARRAY [0..179] OF CHAR;
1041:  BEGIN
1042:    WITH USERINFO DO
1043:      IF (ERRSYM <> SYMCURSOR) OR (ERRBLK <> SYMBLK) THEN
1044:        BEGIN ERRBLK := SYMBLK;
1045:          ERRSYM := SYMCURSOR; ERRNUM := ERRORNUM;
1046:          IF NOISY THEN WRITELN(OUTPUT)
1047:          ELSE
1048:            IF LIST AND (ERRORNUM <= 400) THEN
1049:              EXIT(ERROR);
1050:          IF LINESTART = 0 THEN
1051:            WRITE(OUTPUT,SYMBUFP^:SYMCURSOR)
1052:          ELSE
1053:            BEGIN
1054:              ERRSTART := SCAN(-(LINESTART-1),=CHR(EOL),
1055:                               SYMBUFP^[LINESTART-2])+LINESTART-1;
1056:              MOVELEFT(SYMBUFP^[ERRSTART],A[0],SYMCURSOR-ERRSTART);
```

```
1057:             WRITE(OUTPUT,A:SYMCURSOR-ERRSTART)
1058:           END;
1059:         WRITELN(OUTPUT,' <<<<');
1060:         WRITE(OUTPUT,'LINE ',SCREENDOTS,', ERROR ',ERRORNUM:0,':');
1061:         IF NOISY THEN
1062:           WRITE(OUTPUT,' <SP>(CONTINUE), <ESC>(TERMINATE), E(DIT');
1063:         REPEAT READ(KEYBOARD,CH)
1064:         UNTIL (CH = ' ') OR (CH = 'E') OR (CH = ALTMODE);
1065:         IF CH = 'E' THEN
1066:           BEGIN ERRBLK := SYMBLK-2; EXIT(PASCALCOMPILER) END; (* <<<< SMF *)
1067:         IF (ERRORNUM > 400) OR (CH = CHR(27)) THEN
1068:           BEGIN ERRBLK := 0; EXIT(PASCALCOMPILER) END; (* <<<< SMF 2-25-78 *)
1069:         WRITELN(OUTPUT);
1070:         IF NOISY THEN
1071:           WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
1072:       END
1073:  END (*ERROR*) ;
1074:
1075:  PROCEDURE GETNEXTPAGE;
1076:  BEGIN SYMCURSOR := 0; LINESTART := 0;
1077:    IF INCLUDING THEN
1078:      IF BLOCKREAD(INCLFILE,SYMBUFP^,2,SYMBLK) <> 2 THEN
1079:        BEGIN CLOSE(INCLFILE); INCLUDING := FALSE;
1080:          SYMBLK := OLDSYMBLK; SYMCURSOR := OLDSYMCURSOR;
1081:          LINESTART := OLDLINESTART
1082:        END;
1083:    IF NOT INCLUDING THEN
1084:      IF BLOCKREAD(USERINFO.WORKSYM^,SYMBUFP^,2,SYMBLK) <> 2 THEN
1085:        ERROR(401);
1086:    IF SYMCURSOR = 0 THEN
1087:      IF SYMBUFP^[0] = CHR(16(*DLE*)) THEN
1088:        SYMCURSOR := 2;
1089:    SYMBLK := SYMBLK+2
1090:  END (*GETNEXTPAGE*) ;
1091:
1092:  PROCEDURE PRINTLINE;
1093:    VAR DORC,STARORC: CHAR; LENG: INTEGER;
1094:        A: PACKED ARRAY [0..99] OF CHAR;
1095:  BEGIN DORC := 'C'; STARORC := ':';
1096:    IF DP THEN DORC := 'D';
1097:    IF BPTONLINE THEN STARORC := '*';
1098:    WRITE(LP,SCREENDOTS:6,SEG:4,CURPROC:5,
1099:              STARORC,DORC,LINEINFO:6,' ');
1100:    LENG := SYMCURSOR-LINESTART;
1101:    IF LENG > 100 THEN LENG := 100;
1102:    MOVELEFT(SYMBUFP^[LINESTART],A,LENG);
1103:    IF A[0] = CHR(16(*DLE*)) THEN
1104:      BEGIN
1105:        IF A[1] > ' ' THEN
1106:          WRITE(LP,' ':ORD(A[1])-ORD(' '));
1107:        LENG := LENG-2;
1108:        MOVELEFT(A[2],A,LENG)
1109:      END;
1110:    A[LENG-1] := CHR(EOL); (*JUST TO MAKE SURE*)
1111:    WRITE(LP,A:LENG);
1112:    WITH USERINFO DO
1113:      IF (ERRBLK = SYMBLK) AND (ERRSYM > LINESTART) THEN
1114:        WRITELN(LP,'>>>>>> ERROR # ',ERRNUM)
1115:  END (*PRINTLINE*) ;
1116:
1117:  PROCEDURE ENTERID(FCP: CTP);
1118:    VAR LCP,LCP1: CTP; I: INTEGER;
1119:  BEGIN LCP := DISPLAY[TOP].FNAME;
1120:    IF LCP = NIL THEN DISPLAY[TOP].FNAME := FCP
1121:    ELSE
1122:      BEGIN I := TREESEARCH(LCP,LCP1,FCP^.NAME);
```

```
1123:       WHILE I = 0 DO
1124:         BEGIN ERROR(101);
1125:           IF LCP1^.RLINK = NIL THEN I := 1
1126:           ELSE I := TREESEARCH(LCP1^.RLINK,LCP1,FCP^.NAME)
1127:         END;
1128:       IF I = 1 THEN LCP1^.RLINK := FCP ELSE LCP1^.LLINK := FCP
1129:     END;
1130:   FCP^.LLINK := NIL; FCP^.RLINK := NIL
1131: END (*ENTERID*) ;
1132:
1133: PROCEDURE INSYMBOL; (* COMPILER VERSION 3.4 06-NOV-76 *)
1134:   LABEL 1;
1135:   VAR LVP: CSP; X: INTEGER;
1136:
1137: PROCEDURE CHECKEND;
1138: BEGIN (* CHECKS FOR THE END OF THE PAGE *)
1139:   SCREENDOTS := SCREENDOTS+1;
1140:   SYMCURSOR := SYMCURSOR + 1;
1141:   IF NOISY THEN
1142:     BEGIN WRITE(OUTPUT,'.');
1143:       IF (SCREENDOTS-STARTDOTS) MOD 50 = 0 THEN
1144:         BEGIN WRITELN(OUTPUT);
1145:           WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
1146:         END
1147:     END;
1148:   IF LIST THEN PRINTLINE;
1149:   BPTONLINE := FALSE;
1150:   IF SYMBUFP^[SYMCURSOR]=CHR(0) THEN GETNEXTPAGE
1151:   ELSE LINESTART := SYMCURSOR;
1152:   IF SYMBUFP^[SYMCURSOR] = CHR(12(*FF*)) THEN SYMCURSOR:=SYMCURSOR+1;
1153:   IF SYMBUFP^[SYMCURSOR] = CHR(16(*DLE*)) THEN
1154:     SYMCURSOR := SYMCURSOR+2
1155:   ELSE
1156:     BEGIN
1157:       SYMCURSOR := SYMCURSOR+SCAN(80,<>CHR(9),SYMBUFP^[SYMCURSOR]);
1158:       SYMCURSOR := SYMCURSOR+SCAN(80,<>' ',SYMBUFP^[SYMCURSOR])
1159:     END;
1160:   IF DP THEN LINEINFO := LC ELSE LINEINFO := IC
1161: END;
1162:
1163: PROCEDURE COMMENTER(STOPPER: CHAR);
1164:   VAR CH,SW,DEL: CHAR; LTITLE: STRING[40];
1165:
1166:   PROCEDURE SCANTITLE;
1167:     VAR LENG: INTEGER;
1168:   BEGIN SYMCURSOR := SYMCURSOR+2;
1169:     LENG := SCAN(40,=STOPPER,SYMBUFP^[SYMCURSOR]);
1170:     LTITLE[0] := CHR(LENG);
1171:     MOVELEFT(SYMBUFP^[SYMCURSOR],LTITLE[1],LENG);
1172:     SYMCURSOR := SYMCURSOR+LENG+1
1173:   END (*SCANTITLE*) ;
1174:
1175: BEGIN
1176:   SYMCURSOR := SYMCURSOR+1; (* POINT TO THE FIRST CH PAST "(*" *)
1177:   IF SYMBUFP^[SYMCURSOR]='$' THEN
1178:      IF SYMBUFP^[SYMCURSOR+1] <> STOPPER THEN
1179:         REPEAT
1180:           CH := SYMBUFP^[SYMCURSOR+1];
1181:           SW := SYMBUFP^[SYMCURSOR+2];
1182:           DEL := SYMBUFP^[SYMCURSOR+3];
1183:           IF (SW = ',') OR (SW = STOPPER) THEN
1184:             BEGIN DEL := SW; SW := '+';
1185:               SYMCURSOR := SYMCURSOR-1
1186:             END;
1187:           CASE CH OF
1188:           'D': DEBUGGING := (SW='+');
```

```
1189:           'G': GOTOOK := (SW='+');
1190:           'I': IF (SW='+') OR (SW='-') THEN IOCHECK := (SW='+')
1191:                ELSE
1192:                  BEGIN SCANTITLE;
1193:                    IF STOPPER = '*' THEN
1194:                      SYMCURSOR := SYMCURSOR+1;
1195:                    OPENOLD(INCLFILE,LTITLE);
1196:                    IF IORESULT <> 0 THEN
1197:                      BEGIN OPENOLD(INCLFILE,CONCAT(LTITLE,'.TEXT'));
1198:                         IF IORESULT <> 0 THEN ERROR(403)
1199:                      END;
1200:                    INCLUDING := TRUE;
1201:                    OLDSYMCURSOR := SYMCURSOR;
1202:                    OLDLINESTART := LINESTART;
1203:                    OLDSYMBLK := SYMBLK-2;
1204:                    SYMBLK := 2; GETNEXTPAGE;
1205:                    INSYMBOL; EXIT(INSYMBOL)
1206:                  END;
1207:           'L': IF (SW='+') OR (SW='-') THEN
1208:                  BEGIN LIST := (SW='+');
1209:                    IF LIST THEN OPENNEW(LP,'*SYSTEM.LST.TEXT')
1210:                  END
1211:                ELSE
1212:                  BEGIN SCANTITLE;
1213:                    OPENNEW(LP,LTITLE);
1214:                    LIST := IORESULT = 0;
1215:                    EXIT(COMMENTER)
1216:                  END;
1217:           'Q': NOISY := (SW='-');
1218:           'P': WRITE(LP,CHR(12(*FF*)));
1219:           'R': RANGECHECK := (SW='+');
1220:           'U': BEGIN SYSCOMP := (SW = '-');
1221:                  RANGECHECK := NOT SYSCOMP;
1222:                  IOCHECK := RANGECHECK;
1223:                  GOTOOK := SYSCOMP
1224:                END
1225:           END (*CASES*);
1226:           SYMCURSOR := SYMCURSOR+3;
1227:         UNTIL DEL <> ',';
1228:   SYMCURSOR := SYMCURSOR-1; (* ADJUST *)
1229:   REPEAT
1230:     REPEAT
1231:       SYMCURSOR := SYMCURSOR+1;
1232:       WHILE SYMBUFP^[SYMCURSOR] = CHR(EOL) DO CHECKEND
1233:     UNTIL SYMBUFP^[SYMCURSOR]=STOPPER;
1234:   UNTIL (SYMBUFP^[SYMCURSOR+1]=')') OR (STOPPER='');
1235:   SYMCURSOR := SYMCURSOR+1;
1236: END (*COMMENTER*);
1237:
1238: PROCEDURE STRING;
1239: LABEL 1;
1240: VAR
1241:   T: PACKED ARRAY [1..80] OF CHAR;
1242:   TP,NBLANKS,L: INTEGER;
1243:   DUPLE: BOOLEAN;
1244:
1245: BEGIN
1246:   DUPLE := FALSE; (* INDICATES WHEN '' IS PRESENT *)
1247:   TP := 0; (* INDEX INTO TEMPORARY STRING *)
1248:   REPEAT
1249:     IF DUPLE THEN SYMCURSOR := SYMCURSOR+1;
1250:     REPEAT
1251:       SYMCURSOR := SYMCURSOR+1;
1252:       TP := TP+1;
1253:       IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
1254:         BEGIN ERROR(202); CHECKEND; GOTO 1 END;
```

```
1255:        T[TP] := SYMBUFP^[SYMCURSOR];
1256:      UNTIL SYMBUFP^[SYMCURSOR]='''';
1257:      DUPLE := TRUE;
1258:    UNTIL SYMBUFP^[SYMCURSOR+1]<>'''';
1259:  1:  TP := TP-1; (* ADJUST *)
1260:    SY := STRINGCONST; OP := NOOP;
1261:    LGTH := TP; (* GROSS *)
1262:    IF TP=1 (* SINGLE CHARACTER CONSTANT *)
1263:      THEN
1264:        VAL.IVAL := ORD(T[1])
1265:      ELSE
1266:        WITH SCONST^ DO
1267:          BEGIN
1268:            CCLASS := STRG;
1269:            SLGTH := TP;
1270:            MOVELEFT(T[1],SVAL[1],TP);
1271:            VAL.VALP := SCONST
1272:          END
1273:  END(*STRING*);
1274:
1275:  PROCEDURE NUMBER;
1276:  VAR
1277:    EXPONENT,ENDI,ENDF,ENDE,SIGN,IPART,FPART,EPART,
1278:    ISUM:  INTEGER;
1279:    TIPE: (REALTIPE,INTEGERTIPE);
1280:    RSUM: REAL;
1281:    J: INTEGER;
1282:  BEGIN
1283:    (* TAKES A NUMBER AND DECIDES WHETHER IT'S REAL
1284:       OR INTEGER AND CONVERTS IT TO THE INTERNAL
1285:       FORM. *)
1286:    TIPE := INTEGERTIPE;
1287:    ENDI := 0;
1288:    ENDF := 0;
1289:    ENDE := 0;
1290:    SIGN := 1;
1291:    EPART := 9999; (* OUT OF REACH *)
1292:    IPART := SYMCURSOR; (* INTEGER PART STARTS HERE *)
1293:    REPEAT
1294:      SYMCURSOR := SYMCURSOR+1
1295:    UNTIL (SYMBUFP^[SYMCURSOR]<'0') OR (SYMBUFP^[SYMCURSOR]>'9');
1296:    (* SYMCURSOR NOW POINTS AT FIRST CHARACTER PAST INTEGER PART *)
1297:    ENDI := SYMCURSOR-1; (* MARK THE END OF IPART *)
1298:    IF SYMBUFP^[SYMCURSOR]='.'
1299:      THEN
1300:        IF SYMBUFP^[SYMCURSOR+1]<>'.'  (* WATCH OUT FOR '..' *)
1301:          THEN
1302:            BEGIN
1303:              TIPE := REALTIPE;
1304:              SYMCURSOR := SYMCURSOR+1;
1305:              FPART := SYMCURSOR; (* BEGINNING OF FPART *)
1306:              WHILE (SYMBUFP^[SYMCURSOR] >= '0') AND
1307:                    (SYMBUFP^[SYMCURSOR] <= '9') DO
1308:                SYMCURSOR := SYMCURSOR+1;
1309:              IF SYMCURSOR = FPART THEN ERROR(201);
1310:              ENDF := SYMCURSOR-1;
1311:            END;
1312:    IF SYMBUFP^[SYMCURSOR]='E'
1313:      THEN
1314:        BEGIN
1315:          TIPE := REALTIPE;
1316:          SYMCURSOR := SYMCURSOR+1;
1317:          IF SYMBUFP^[SYMCURSOR]='-'
1318:            THEN
1319:              BEGIN
1320:                SYMCURSOR := SYMCURSOR+1;
```

```
1321:               SIGN := -1;
1322:             END
1323:           ELSE
1324:             IF SYMBUFP^[SYMCURSOR]='+'
1325:               THEN
1326:                 SYMCURSOR := SYMCURSOR+1;
1327:         EPART := SYMCURSOR; (* BEGINNING OF EXPONENT *)
1328:         WHILE (SYMBUFP^[SYMCURSOR]>='0') AND (SYMBUFP^[SYMCURSOR]<='9') DO
1329:           SYMCURSOR := SYMCURSOR+1;
1330:         ENDE := SYMCURSOR-1;
1331:         IF ENDE<EPART THEN ERROR(201); (* ERROR IN REAL CONSTANT *)
1332:       END;
1333:   (* NOW CONVERT TO INTERNAL FORM *)
1334:   IF TIPE=INTEGERTIPE
1335:     THEN
1336:       BEGIN
1337:         ISUM := 0;
1338:         FOR J := IPART TO ENDI DO
1339:           BEGIN
1340:             IF (ISUM>3276) OR ((ISUM=3276) AND (SYMBUFP^[J]>'7')) THEN
1341:                 BEGIN ERROR(203); J := ENDI END
1342:             ELSE ISUM := ISUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
1343:           END;
1344:           SY := INTCONST;  OP := NOOP;
1345:           VAL.IVAL := ISUM;
1346:         END
1347:       ELSE
1348:        BEGIN (* REAL NUMBER HERE *)
1349:          RSUM := 0;
1350:          FOR J := IPART TO ENDI DO
1351:            BEGIN
1352:              RSUM := RSUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
1353:            END;
1354:          FOR J := ENDF DOWNTO FPART DO
1355:            RSUM := RSUM+(ORD(SYMBUFP^[J])-ORD('0'))/PWROFTEN(J-FPART+1);
1356:          EXPONENT := 0;
1357:          FOR J := EPART TO ENDE DO
1358:            EXPONENT := EXPONENT*10+ORD(SYMBUFP^[J])-ORD('0');
1359:          IF SIGN=-1
1360:            THEN
1361:              RSUM := RSUM/PWROFTEN(EXPONENT)
1362:            ELSE
1363:              RSUM := RSUM*PWROFTEN(EXPONENT);
1364:          SY := REALCONST;  OP := NOOP;
1365:          NEW(LVP,REEL);
1366:          LVP^.CCLASS := REEL;
1367:          LVP^.RVAL := RSUM;
1368:          VAL.VALP := LVP;
1369:        END;
1370:   SYMCURSOR := SYMCURSOR-1; (* ADJUST FOR POSTERITY *)
1371: END;
1372:
1373: BEGIN (* INSYMBOL *)
1374:   OP := NOOP;
1375: 1:  SY := OTHERSY; (* IF NO CASES EXERCISED BLOW UP *)
1376:   CASE SYMBUFP^[SYMCURSOR] OF
1377:   '''':STRING;
1378:   '0','1','2','3','4','5','6','7','8','9':
1379:        NUMBER;
1380:   'A','B','C','D','E','F','G','H','I','J','K','L','M',
1381:   'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
1382:   'A','B','C','D','E','F','G','H','I','J','K','L','M',
1383:   'N','O','P','Q','R','S','T','U','V','W','X','Y','Z':
1384:        IDSEARCH(SYMCURSOR,SYMBUFP^); (* MAGIC PROC *)
1385:   '': BEGIN COMMENTER(''); GOTO 1 END;
1386:   '(': BEGIN
```

```
1387:           IF SYMBUFP^[SYMCURSOR+1]='*' THEN
1388:               BEGIN
1389:                  SYMCURSOR := SYMCURSOR+1;
1390:                  COMMENTER('*');
1391:                  SYMCURSOR := SYMCURSOR+1;
1392:                  GOTO 1; (* GET ANOTHER TOKEN *)
1393:               END
1394:             ELSE
1395:               SY := LPARENT;
1396:         END;
1397:     ')': SY := RPARENT;
1398:     ',': SY := COMMA;
1399:     ' ',' ': BEGIN SYMCURSOR := SYMCURSOR+1; GOTO 1; END;
1400:     '.': BEGIN
1401:           IF SYMBUFP^[SYMCURSOR+1]='.'
1402:             THEN
1403:               BEGIN
1404:                 SYMCURSOR := SYMCURSOR+1;
1405:                 SY := COLON
1406:               END
1407:             ELSE
1408:               SY := PERIOD;
1409:         END;
1410:     ':': IF SYMBUFP^[SYMCURSOR+1]='='
1411:           THEN
1412:             BEGIN
1413:               SYMCURSOR := SYMCURSOR+1;
1414:               SY := BECOMES;
1415:              END
1416:           ELSE
1417:              SY := COLON;
1418:     ';': SY := SEMICOLON;
1419:     '^': SY := ARROW;
1420:     '[': SY := LBRACK;
1421:     ']': SY := RBRACK;
1422:     '*': BEGIN SY := MULOP; OP := MUL END;
1423:     '+': BEGIN SY := ADDOP; OP := PLUS END;
1424:     '-': BEGIN SY := ADDOP; OP := MINUS END;
1425:     '/': BEGIN SY := MULOP; OP := RDIV END;
1426:     '<': BEGIN
1427:           SY := RELOP;
1428:           OP := LTOP;
1429:           CASE SYMBUFP^[SYMCURSOR+1] OF
1430:             '>': BEGIN
1431:                    OP := NEOP;
1432:                    SYMCURSOR := SYMCURSOR+1
1433:                  END;
1434:             '=': BEGIN
1435:                    OP := LEOP;
1436:                    SYMCURSOR := SYMCURSOR+1
1437:                  END
1438:           END;
1439:         END;
1440:     '=': BEGIN SY := RELOP; OP := EQOP END;
1441:     '>': BEGIN
1442:           SY := RELOP;
1443:           IF SYMBUFP^[SYMCURSOR+1]='='
1444:             THEN
1445:               BEGIN
1446:                 OP := GEOP;
1447:                 SYMCURSOR := SYMCURSOR+1;
1448:               END
1449:             ELSE
1450:               OP := GTOP;
1451:         END
1452:   END (* CASE SYMBUFP^[SYMCURSOR] OF *);
```

```
1453:    IF SY=OTHERSY THEN
1454:      IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
1455:        BEGIN CHECKEND; GOTO 1 END
1456:      ELSE ERROR(400);
1457:    SYMCURSOR := SYMCURSOR+1; (* NEXT CALL TALKS ABOUT NEXT TOKEN *)
1458:  END (*INSYMBOL*) ;
1459:
1460:
1461:    (*      COPYRIGHT  (C) 1978, REGENTS OF THE      *)
1462:    (*      UNIVERSITY OF CALIFORNIA, SAN DIEGO      *)
1463:
1464:    PROCEDURE SEARCHSECTION(FCP: CTP; VAR FCP1: CTP);
1465:    BEGIN
1466:      IF FCP <> NIL THEN
1467:        IF TREESEARCH(FCP,FCP1,ID) = 0 THEN (*NADA*)
1468:        ELSE FCP1 := NIL
1469:      ELSE FCP1 := NIL
1470:    END (*SEARCHSECTION*) ;
1471:
1472:    PROCEDURE SEARCHID(FIDCLS: SETOFIDS; VAR FCP: CTP);
1473:      LABEL 1; VAR LCP: CTP;
1474:    BEGIN
1475:      FOR DISX := TOP DOWNTO 0 DO
1476:        BEGIN LCP := DISPLAY[DISX].FNAME;
1477:          IF LCP <> NIL THEN
1478:            IF TREESEARCH(LCP,LCP,ID) = 0 THEN
1479:              IF LCP^.KLASS IN FIDCLS THEN GOTO 1
1480:              ELSE
1481:                IF PRTERR THEN ERROR(103)
1482:                ELSE LCP := NIL
1483:            ELSE LCP := NIL
1484:        END;
1485:      IF PRTERR THEN
1486:        BEGIN ERROR(104);
1487:          IF TYPES IN FIDCLS THEN LCP := UTYPPTR
1488:          ELSE
1489:            IF VARS IN FIDCLS THEN LCP := UVARPTR
1490:            ELSE
1491:              IF FIELD IN FIDCLS THEN LCP := UFLDPTR
1492:              ELSE
1493:                IF KONST IN FIDCLS THEN LCP := UCSTPTR
1494:                ELSE
1495:                  IF PROC IN FIDCLS THEN LCP := UPRCPTR
1496:                  ELSE LCP := UFCTPTR
1497:        END;
1498:  1:  FCP := LCP
1499:    END (*SEARCHID*) ;
1500:
1501:    PROCEDURE GETBOUNDS(FSP: STP; VAR FMIN,FMAX: INTEGER);
1502:    BEGIN
1503:      WITH FSP^ DO
1504:        IF FORM = SUBRANGE THEN
1505:          BEGIN FMIN := MIN.IVAL; FMAX := MAX.IVAL END
1506:        ELSE
1507:          BEGIN FMIN := 0;
1508:            IF FSP = CHARPTR THEN FMAX := 255
1509:            ELSE
1510:              IF FSP^.FCONST <> NIL THEN
1511:                FMAX := FSP^.FCONST^.VALUES.IVAL
1512:              ELSE FMAX := 0
1513:          END
1514:    END (*GETBOUNDS*) ;
1515:
1516:    PROCEDURE SKIP(FSYS: SETOFSYS);
1517:    BEGIN WHILE NOT(SY IN FSYS) DO INSYMBOL
1518:    END (*SKIP*) ;
```

```
1519:
1520:    FUNCTION PAOFCHAR(FSP: STP): BOOLEAN;
1521:    BEGIN PAOFCHAR := FALSE;
1522:      IF FSP <> NIL THEN
1523:        IF FSP^.FORM = ARRAYS THEN
1524:          PAOFCHAR := FSP^.AISPACKD AND (FSP^.AELTYPE = CHARPTR)
1525:    END (*PAOFCHAR*) ;
1526:
1527:    FUNCTION STRGTYPE(FSP: STP) : BOOLEAN;
1528:    BEGIN STRGTYPE := FALSE;
1529:      IF PAOFCHAR(FSP) THEN STRGTYPE := FSP^.AISSTRNG
1530:    END (*STRGTYPE*) ;
1531:
1532:    PROCEDURE CONSTANT(FSYS: SETOFSYS; VAR FSP: STP; VAR FVALU: VALU);
1533:      VAR LSP: STP; LCP: CTP; SIGN: (NONE,POS,NEG);
1534:          LVP: CSP;
1535:    BEGIN LSP := NIL; FVALU.IVAL := 0;
1536:      IF NOT(SY IN CONSTBEGSYS) THEN
1537:        BEGIN ERROR(50); SKIP(FSYS+CONSTBEGSYS) END;
1538:      IF SY IN CONSTBEGSYS THEN
1539:        BEGIN
1540:          IF SY = STRINGCONSTSY THEN
1541:            BEGIN
1542:              IF LGTH = 1 THEN LSP := CHARPTR
1543:              ELSE
1544:                BEGIN
1545:                  NEW(LSP,ARRAYS,TRUE,TRUE);
1546:                  LSP^ := STRGPTR^;
1547:                  LSP^.MAXLENG := LGTH;
1548:                  LSP^.INXTYPE := NIL;
1549:                  NEW(LVP);
1550:                  LVP^ := VAL.VALP^;
1551:                  VAL.VALP := LVP
1552:                END;
1553:              FVALU := VAL; INSYMBOL
1554:            END
1555:          ELSE
1556:            BEGIN
1557:              SIGN := NONE;
1558:              IF (SY = ADDOP) AND (OP IN [PLUS,MINUS]) THEN
1559:                BEGIN IF OP = PLUS THEN SIGN := POS ELSE SIGN := NEG;
1560:                  INSYMBOL
1561:                END;
1562:              IF SY = IDENT THEN
1563:                BEGIN SEARCHID([KONST],LCP);
1564:                  WITH LCP^ DO
1565:                    BEGIN LSP := IDTYPE; FVALU := VALUES END;
1566:                  IF SIGN <> NONE THEN
1567:                    IF LSP = INTPTR THEN
1568:                      BEGIN IF SIGN = NEG THEN
1569:                        FVALU.IVAL := -FVALU.IVAL END
1570:                    ELSE
1571:                      IF LSP = REALPTR THEN
1572:                        BEGIN
1573:                          IF SIGN = NEG THEN
1574:                            BEGIN NEW(LVP,REEL);
1575:                              LVP^.CCLASS := REEL;
1576:                              LVP^.RVAL := -FVALU.VALP^.RVAL;
1577:                              FVALU.VALP := LVP;
1578:                            END
1579:                        END
1580:                      ELSE ERROR(105);
1581:                  INSYMBOL;
1582:                END
1583:              ELSE
1584:                IF SY = INTCONST THEN
```

```
1585:                 BEGIN IF SIGN = NEG THEN VAL.IVAL := -VAL.IVAL;
1586:                   LSP := INTPTR; FVALU := VAL; INSYMBOL
1587:                 END
1588:               ELSE
1589:                 IF SY = REALCONST THEN
1590:                   BEGIN IF SIGN = NEG THEN
1591:                           VAL.VALP^.RVAL := -VAL.VALP^.RVAL;
1592:                     LSP := REALPTR; FVALU := VAL; INSYMBOL
1593:                   END
1594:                 ELSE
1595:                   BEGIN ERROR(106); SKIP(FSYS) END
1596:           END;
1597:         IF NOT (SY IN FSYS) THEN
1598:           BEGIN ERROR(6); SKIP(FSYS) END
1599:         END;
1600:     FSP := LSP
1601:   END (*CONSTANT*) ;
1602:
1603:   FUNCTION COMPTYPES(FSP1,FSP2: STP) : BOOLEAN;
1604:     VAR NXT1,NXT2: CTP; COMP: BOOLEAN;
1605:       LTESTP1,LTESTP2 : TESTP;
1606:   BEGIN
1607:     IF FSP1 = FSP2 THEN COMPTYPES := TRUE
1608:     ELSE
1609:       IF (FSP1 = NIL) OR (FSP2 = NIL) THEN COMPTYPES := TRUE
1610:       ELSE
1611:         IF FSP1^.FORM = FSP2^.FORM THEN
1612:           CASE FSP1^.FORM OF
1613:             SCALAR:
1614:               COMPTYPES := FALSE;
1615:             SUBRANGE:
1616:               COMPTYPES := COMPTYPES(FSP1^.RANGETYPE,
1617:                                      FSP2^.RANGETYPE);
1618:             POINTER:
1619:                 BEGIN
1620:                   COMP := FALSE; LTESTP1 := GLOBTESTP;
1621:                   LTESTP2 := GLOBTESTP;
1622:                   WHILE LTESTP1 <> NIL DO
1623:                     WITH LTESTP1^ DO
1624:                       BEGIN
1625:                         IF (ELT1 = FSP1^.ELTYPE) AND
1626:                           (ELT2 = FSP2^.ELTYPE) THEN COMP := TRUE;
1627:                         LTESTP1 := LASTTESTP
1628:                       END;
1629:                   IF NOT COMP THEN
1630:                     BEGIN NEW(LTESTP1);
1631:                       WITH LTESTP1^ DO
1632:                         BEGIN ELT1 := FSP1^.ELTYPE;
1633:                           ELT2 := FSP2^.ELTYPE;
1634:                           LASTTESTP := GLOBTESTP
1635:                         END;
1636:                       GLOBTESTP := LTESTP1;
1637:                       COMP := COMPTYPES(FSP1^.ELTYPE,FSP2^.ELTYPE)
1638:                     END;
1639:                   COMPTYPES := COMP; GLOBTESTP := LTESTP2
1640:                 END;
1641:             POWER:
1642:               COMPTYPES := COMPTYPES(FSP1^.ELSET,FSP2^.ELSET);
1643:             ARRAYS:
1644:               BEGIN
1645:                 COMP := COMPTYPES(FSP1^.AELTYPE,FSP2^.AELTYPE)
1646:                         AND (FSP1^.AISPACKD = FSP2^.AISPACKD);
1647:                 IF COMP AND FSP1^.AISPACKD THEN
1648:                     COMP := (FSP1^.ELSPERWD = FSP2^.ELSPERWD)
1649:                             AND (FSP1^.ELWIDTH = FSP2^.ELWIDTH)
1650:                             AND (FSP1^.AISSTRNG = FSP2^.AISSTRNG);
```

```
1651:                 IF COMP AND NOT STRGTYPE(FSP1) THEN
1652:                   COMP := (FSP1^.SIZE = FSP2^.SIZE);
1653:                 COMPTYPES := COMP;
1654:               END;
1655:             RECORDS:
1656:               BEGIN NXT1 := FSP1^.FSTFLD; NXT2 := FSP2^.FSTFLD;
1657:                 COMP := TRUE;
1658:                 WHILE (NXT1 <> NIL) AND (NXT2 <> NIL) AND COMP DO
1659:                   BEGIN COMP:=COMPTYPES(NXT1^.IDTYPE,NXT2^.IDTYPE);
1660:                     NXT1 := NXT1^.NEXT; NXT2 := NXT2^.NEXT
1661:                   END;
1662:                 COMPTYPES := COMP AND (NXT1 = NIL) AND (NXT2 = NIL)
1663:                           AND (FSP1^.RECVAR = NIL)
1664:                           AND (FSP2^.RECVAR = NIL)
1665:               END;
1666:             FILES:
1667:               COMPTYPES := COMPTYPES(FSP1^.FILTYPE,FSP2^.FILTYPE)
1668:           END (*CASE*)
1669:         ELSE (*FSP1^.FORM <> FSP2^.FORM*)
1670:           IF FSP1^.FORM = SUBRANGE THEN
1671:             COMPTYPES := COMPTYPES(FSP1^.RANGETYPE,FSP2)
1672:           ELSE
1673:             IF FSP2^.FORM = SUBRANGE THEN
1674:               COMPTYPES := COMPTYPES(FSP1,FSP2^.RANGETYPE)
1675:             ELSE COMPTYPES := FALSE
1676:   END (*COMPTYPES*) ;
1677:  (*    COPYRIGHT (C) 1978, REGENTS OF THE       *)
1678: (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
1679:
1680:   PROCEDURE TYP(FSYS: SETOFSYS; VAR FSP: STP; VAR FSIZE: ADDRRANGE);
1681:     VAR LSP,LSP1,LSP2: STP; OLDTOP: DISPRANGE; LCP: CTP;
1682:         LSIZE,DISPL: ADDRRANGE; LMIN,LMAX: INTEGER;
1683:         PACKING: BOOLEAN; NEXTBIT,NUMBITS: BITRANGE;
1684:
1685:     PROCEDURE SIMPLETYPE(FSYS:SETOFSYS; VAR FSP:STP; VAR FSIZE:ADDRRANGE);
1686:       VAR LSP,LSP1: STP; LCP,LCP1: CTP; TTOP: DISPRANGE;
1687:           LCNT: INTEGER; LVALU: VALU;
1688:     BEGIN FSIZE := 1;
1689:       IF NOT (SY IN SIMPTYPEBEGSYS) THEN
1690:         BEGIN ERROR(1); SKIP(FSYS + SIMPTYPEBEGSYS) END;
1691:       IF SY IN SIMPTYPEBEGSYS THEN
1692:         BEGIN
1693:           IF SY = LPARENT THEN
1694:             BEGIN TTOP := TOP;
1695:               WHILE DISPLAY[TOP].OCCUR <> BLCK DO TOP := TOP - 1;
1696:               NEW(LSP,SCALAR,DECLARED);
1697:               WITH LSP^ DO
1698:                 BEGIN SIZE := INTSIZE; FORM := SCALAR;
1699:                   SCALKIND := DECLARED
1700:                 END;
1701:               LCP1 := NIL; LCNT := 0;
1702:               REPEAT INSYMBOL;
1703:                 IF SY = IDENT THEN
1704:                   BEGIN NEW(LCP,KONST);
1705:                     WITH LCP^ DO
1706:                       BEGIN NAME := ID; IDTYPE := LSP; NEXT := LCP1;
1707:                         VALUES.IVAL := LCNT; KLASS := KONST
1708:                       END;
1709:                     ENTERID(LCP);
1710:                     LCNT := LCNT + 1;
1711:                     LCP1 := LCP; INSYMBOL
1712:                   END
1713:                 ELSE ERROR(2);
1714:                 IF NOT (SY IN FSYS + [COMMA,RPARENT]) THEN
1715:                   BEGIN ERROR(6); SKIP(FSYS + [COMMA,RPARENT]) END
1716:               UNTIL SY <> COMMA;
```

```
1717:                LSP^.FCONST := LCP1; TOP := TTOP;
1718:                IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
1719:              END
1720:            ELSE
1721:              BEGIN
1722:                IF SY = IDENT THEN
1723:                  BEGIN SEARCHID([TYPES,KONST],LCP);
1724:                    INSYMBOL;
1725:                    IF LCP^.KLASS = KONST THEN
1726:                      BEGIN NEW(LSP,SUBRANGE);
1727:                        WITH LSP^, LCP^ DO
1728:                          BEGIN RANGETYPE := IDTYPE; FORM := SUBRANGE;
1729:                            IF STRGTYPE(RANGETYPE) THEN
1730:                              BEGIN ERROR(148); RANGETYPE := NIL END;
1731:                            MIN := VALUES; SIZE := INTSIZE
1732:                          END;
1733:                        IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1734:                        CONSTANT(FSYS,LSP1,LVALU);
1735:                        LSP^.MAX := LVALU;
1736:                        IF LSP^.RANGETYPE <> LSP1 THEN ERROR(107)
1737:                      END
1738:                    ELSE
1739:                      BEGIN LSP := LCP^.IDTYPE;
1740:                        IF (LSP = STRGPTR) AND (SY = LBRACK) THEN
1741:                          BEGIN INSYMBOL;
1742:                            CONSTANT(FSYS + [RBRACK],LSP1,LVALU);
1743:                            IF LSP1 = INTPTR THEN
1744:                              BEGIN
1745:                                IF (LVALU.IVAL <= 0) OR
1746:                                   (LVALU.IVAL > STRGLGTH) THEN
1747:                                  BEGIN ERROR(203);
1748:                                    LVALU.IVAL := DEFSTRGLGTH
1749:                                  END;
1750:                                IF LVALU.IVAL <> DEFSTRGLGTH THEN
1751:                                  BEGIN NEW(LSP,ARRAYS,TRUE,TRUE);
1752:                                    LSP^ := STRGPTR^;
1753:                                    WITH LSP^,LVALU DO
1754:                                      BEGIN MAXLENG := IVAL;
1755:                                        SIZE := (IVAL+CHRSPERWD) DIV CHRSPERWD
1756:                                      END
1757:                                  END
1758:                              END
1759:                            ELSE ERROR(15);
1760:                            IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
1761:                          END;
1762:                        IF LSP <> NIL THEN FSIZE := LSP^.SIZE
1763:                      END
1764:                  END (*SY = IDENT*)
1765:                ELSE
1766:                  BEGIN NEW(LSP,SUBRANGE); LSP^.FORM := SUBRANGE;
1767:                    CONSTANT(FSYS + [COLON],LSP1,LVALU);
1768:                    IF STRGTYPE(LSP1) THEN
1769:                      BEGIN ERROR(148); LSP1 := NIL END;
1770:                    WITH LSP^ DO
1771:                      BEGIN RANGETYPE:=LSP1; MIN:=LVALU; SIZE:=INTSIZE END;
1772:                    IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1773:                    CONSTANT(FSYS,LSP1,LVALU);
1774:                    LSP^.MAX := LVALU;
1775:                    IF LSP^.RANGETYPE <> LSP1 THEN ERROR(107)
1776:                  END;
1777:                IF LSP <> NIL THEN
1778:                  WITH LSP^ DO
1779:                    IF FORM = SUBRANGE THEN
1780:                      IF RANGETYPE <> NIL THEN
1781:                        IF RANGETYPE = REALPTR THEN ERROR(399)
1782:                        ELSE
```

```
1783:                         IF MIN.IVAL > MAX.IVAL THEN
1784:                           BEGIN ERROR(102); MAX.IVAL := MIN.IVAL END
1785:             END;
1786:           FSP := LSP;
1787:           IF NOT (SY IN FSYS) THEN
1788:             BEGIN ERROR(6); SKIP(FSYS) END
1789:         END
1790:           ELSE FSP := NIL
1791:     END (*SIMPLETYPE*) ;
1792:
1793:     FUNCTION PACKABLE(FSP: STP): BOOLEAN;
1794:       VAR LMIN,LMAX: INTEGER;
1795:     BEGIN PACKABLE := FALSE;
1796:       IF (FSP <> NIL) AND PACKING THEN
1797:         WITH FSP^ DO
1798:           CASE FORM OF
1799:             SUBRANGE,
1800:             SCALAR:  IF (FSP <> INTPTR) AND (FSP <> REALPTR) THEN
1801:                        BEGIN GETBOUNDS(FSP,LMIN,LMAX);
1802:                          IF LMIN >= 0 THEN
1803:                            BEGIN PACKABLE := TRUE;
1804:                              NUMBITS := 1; LMIN := 1;
1805:                              WHILE LMIN < LMAX DO
1806:                                BEGIN LMIN := LMIN + 1;
1807:                                  LMIN := LMIN + LMIN - 1;
1808:                                  NUMBITS := NUMBITS + 1
1809:                                END
1810:                            END
1811:                        END;
1812:             POWER:   IF PACKABLE(ELSET) THEN
1813:                        BEGIN GETBOUNDS(ELSET,LMIN,LMAX);
1814:                          LMAX := LMAX + 1;
1815:                          IF LMAX < BITSPERWD THEN
1816:                            BEGIN PACKABLE := TRUE;
1817:                              NUMBITS := LMAX
1818:                            END
1819:                        END
1820:           END (* CASES *);
1821:     END (*PACKABLE*) ;
1822:
1823:     PROCEDURE FIELDLIST(FSYS: SETOFSYS; VAR FRECVAR: STP);
1824:       VAR LCP,LCP1,NXT,NXT1,LAST: CTP; LSP,LSP1,LSP2,LSP3,LSP4: STP;
1825:           MINSIZE,MAXSIZE,LSIZE: ADDRRANGE; LVALU: VALU;
1826:           MAXBIT,MINBIT: BITRANGE;
1827:
1828:       PROCEDURE ALLOCATE(FCP: CTP);
1829:         VAR ONBOUND: BOOLEAN;
1830:       BEGIN ONBOUND := FALSE;
1831:         WITH FCP^ DO
1832:           IF PACKABLE(IDTYPE) THEN
1833:             BEGIN
1834:               IF (NUMBITS + NEXTBIT) > BITSPERWD THEN
1835:                 BEGIN DISPL := DISPL + 1; NEXTBIT := 0; ONBOUND := TRUE END;
1836:               FLDADDR := DISPL; FISPACKD := TRUE;
1837:               FLDWIDTH := NUMBITS; FLDRBIT := NEXTBIT;
1838:               NEXTBIT := NEXTBIT + NUMBITS
1839:             END
1840:           ELSE
1841:             BEGIN DISPL := DISPL + ORD(NEXTBIT > 0);
1842:               NEXTBIT := 0; ONBOUND := TRUE;
1843:               FISPACKD := FALSE; FLDADDR := DISPL;
1844:               IF IDTYPE <> NIL THEN
1845:                 DISPL := DISPL + IDTYPE^.SIZE
1846:             END;
1847:         IF ONBOUND AND (LAST <> NIL) THEN
1848:           WITH LAST^ DO
```

```
1849:             IF FISPACKD THEN
1850:               IF FLDRBIT = 0 THEN FISPACKD := FALSE
1851:               ELSE
1852:                 IF (FLDWIDTH <= 8) AND (FLDRBIT <= 8) THEN
1853:                   BEGIN FLDWIDTH := 8; FLDRBIT := 8 END
1854:       END (*ALLOCATE*) ;
1855:
1856:       PROCEDURE VARIANTLIST;
1857:         VAR GOTTAGNAME: BOOLEAN;
1858:       BEGIN NEW(LSP,TAGFLD);
1859:         WITH LSP^ DO
1860:           BEGIN TAGFIELDP := NIL; FSTVAR := NIL; FORM := TAGFLD END;
1861:         FRECVAR := LSP;
1862:         INSYMBOL;
1863:         IF SY = IDENT THEN
1864:           BEGIN
1865:             IF PACKING THEN NEW(LCP,FIELD,TRUE)
1866:             ELSE NEW(LCP,FIELD,FALSE);
1867:             WITH LCP^ DO
1868:               BEGIN IDTYPE := NIL; KLASS:=FIELD;
1869:                 NEXT := NIL; FISPACKD := FALSE
1870:               END;
1871:             GOTTAGNAME := FALSE; PRTERR := FALSE;
1872:             SEARCHID([TYPES],LCP1); PRTERR := TRUE;
1873:             IF LCP1 = NIL THEN
1874:               BEGIN GOTTAGNAME := TRUE;
1875:                 LCP^.NAME := ID; ENTERID(LCP); INSYMBOL;
1876:                 IF SY = COLON THEN INSYMBOL ELSE ERROR(5)
1877:               END;
1878:             IF SY = IDENT THEN
1879:               BEGIN SEARCHID([TYPES],LCP1);
1880:                 LSP1 := LCP1^.IDTYPE;
1881:                 IF LSP1 <> NIL THEN
1882:                   BEGIN
1883:                     IF LSP1^.FORM <= SUBRANGE THEN
1884:                       BEGIN
1885:                         IF COMPTYPES(REALPTR,LSP1) THEN ERROR(109);
1886:                         LCP^.IDTYPE := LSP1; LSP^.TAGFIELDP := LCP;
1887:                         IF GOTTAGNAME THEN ALLOCATE(LCP)
1888:                       END
1889:                     ELSE ERROR(110)
1890:                   END;
1891:                 INSYMBOL
1892:               END
1893:             ELSE BEGIN ERROR(2); SKIP(FSYS + [OFSY,LPARENT]) END
1894:           END
1895:         ELSE BEGIN ERROR(2); SKIP(FSYS + [OFSY,LPARENT]) END;
1896:         LSP^.SIZE := DISPL + ORD(NEXTBIT > 0);
1897:         IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1898:         LSP1 := NIL; MINSIZE := DISPL; MAXSIZE := DISPL;
1899:         MINBIT := NEXTBIT; MAXBIT := NEXTBIT;
1900:         REPEAT LSP2 := NIL;
1901:           REPEAT CONSTANT(FSYS + [COMMA,COLON,LPARENT],LSP3,LVALU);
1902:             IF LSP^.TAGFIELDP <> NIL THEN
1903:               IF NOT COMPTYPES(LSP^.TAGFIELDP^.IDTYPE,LSP3) THEN
1904:                 ERROR(111);
1905:             NEW(LSP3,VARIANT);
1906:             WITH LSP3^ DO
1907:               BEGIN NXTVAR := LSP1; SUBVAR := LSP2;
1908:                 VARVAL := LVALU; FORM := VARIANT
1909:               END;
1910:             LSP1 := LSP3; LSP2 := LSP3;
1911:             TEST := SY <> COMMA;
1912:             IF NOT TEST THEN INSYMBOL
1913:           UNTIL TEST;
1914:           IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
```

```
1915:           IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
1916:           IF SY = RPARENT THEN LSP2 := NIL
1917:           ELSE
1918:             FIELDLIST(FSYS + [RPARENT,SEMICOLON],LSP2);
1919:           IF DISPL > MAXSIZE THEN
1920:             BEGIN MAXSIZE := DISPL; MAXBIT := NEXTBIT END
1921:           ELSE
1922:             IF (DISPL = MAXSIZE) AND (NEXTBIT > MAXBIT) THEN
1923:               MAXBIT := NEXTBIT;
1924:           WHILE LSP3 <> NIL DO
1925:             BEGIN LSP4 := LSP3^.SUBVAR; LSP3^.SUBVAR := LSP2;
1926:               LSP3^.SIZE := DISPL + ORD(NEXTBIT > 0);
1927:               LSP3 := LSP4
1928:             END;
1929:           IF SY = RPARENT THEN
1930:             BEGIN INSYMBOL;
1931:               IF NOT (SY IN FSYS + [SEMICOLON]) THEN
1932:                 BEGIN ERROR(6); SKIP(FSYS + [SEMICOLON]) END
1933:             END
1934:           ELSE ERROR(4);
1935:           TEST := SY <> SEMICOLON;
1936:           IF NOT TEST THEN
1937:             BEGIN INSYMBOL;
1938:               DISPL := MINSIZE; NEXTBIT := MINBIT
1939:             END
1940:         UNTIL (TEST) OR (SY = ENDSY); (* <<<< SMF 2-28-78 *)
1941:         DISPL := MAXSIZE; NEXTBIT := MAXBIT;
1942:         LSP^.FSTVAR := LSP1
1943:       END (*VARIANTLIST*) ;
1944:
1945:     BEGIN (*FIELDLIST*)
1946:       NXT1 := NIL; LSP := NIL; LAST := NIL;
1947:       IF NOT (SY IN [IDENT,CASESY]) THEN
1948:         BEGIN ERROR(19); SKIP(FSYS + [IDENT,CASESY]) END;
1949:       WHILE SY = IDENT DO
1950:         BEGIN NXT := NXT1;
1951:           REPEAT
1952:             IF SY = IDENT THEN
1953:               BEGIN
1954:                 IF PACKING THEN NEW(LCP,FIELD,TRUE)
1955:                 ELSE NEW(LCP,FIELD,FALSE);
1956:                 WITH LCP^ DO
1957:                   BEGIN NAME := ID; IDTYPE := NIL; NEXT := NXT;
1958:                     KLASS := FIELD; FISPACKD := FALSE
1959:                   END;
1960:                 NXT := LCP;
1961:                 ENTERID(LCP);
1962:                 INSYMBOL
1963:               END
1964:             ELSE ERROR(2);
1965:             IF NOT (SY IN [COMMA,COLON]) THEN
1966:               BEGIN ERROR(6); SKIP(FSYS + [COMMA,COLON,SEMICOLON,CASESY]) END;
1967:             TEST := SY <> COMMA;
1968:             IF NOT TEST  THEN INSYMBOL
1969:           UNTIL TEST;
1970:           IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1971:           TYP(FSYS + [CASESY,SEMICOLON],LSP,LSIZE);
1972:           IF LSP <> NIL THEN
1973:             IF LSP^.FORM = FILES THEN ERROR(108);
1974:           WHILE NXT <> NXT1 DO
1975:             WITH NXT^ DO
1976:               BEGIN IDTYPE := LSP; ALLOCATE(NXT);
1977:                 IF NEXT = NXT1 THEN LAST := NXT;
1978:                 NXT := NEXT
1979:               END;
1980:           NXT1 := LCP;
```

```
1981:            IF SY = SEMICOLON THEN
1982:              BEGIN INSYMBOL;
1983:                IF NOT (SY IN [IDENT,ENDSY,CASESY]) THEN (* <<<< SMF 2-28-78 *)
1984:                  BEGIN ERROR(19); SKIP(FSYS + [IDENT,CASESY]) END
1985:              END
1986:          END (*WHILE*);
1987:        NXT := NIL;
1988:        WHILE NXT1 <> NIL DO
1989:          WITH NXT1^ DO
1990:            BEGIN LCP := NEXT; NEXT := NXT; NXT := NXT1; NXT1 := LCP END;
1991:        IF SY = CASESY THEN VARIANTLIST
1992:        ELSE FRECVAR := NIL
1993:      END (*FIELDLIST*) ;
1994:
1995:      PROCEDURE POINTERTYPE;
1996:      BEGIN NEW(LSP,POINTER); FSP := LSP;
1997:        WITH LSP^ DO
1998:          BEGIN ELTYPE := NIL; SIZE := PTRSIZE; FORM := POINTER END;
1999:        INSYMBOL;
2000:        IF SY = IDENT THEN
2001:          BEGIN PRTERR := FALSE;
2002:            SEARCHID([TYPES],LCP); PRTERR := TRUE;
2003:            IF LCP = NIL THEN   (*FORWARD REFERENCED TYPE ID*)
2004:              BEGIN NEW(LCP,TYPES);
2005:                WITH LCP^ DO
2006:                  BEGIN NAME := ID; IDTYPE := LSP;
2007:                    NEXT := FWPTR; KLASS := TYPES
2008:                  END;
2009:                FWPTR := LCP
2010:              END
2011:            ELSE
2012:              BEGIN
2013:                IF LCP^.IDTYPE <> NIL THEN
2014:                  IF (LCP^.IDTYPE^.FORM <> FILES) OR SYSCOMP THEN
2015:                    LSP^.ELTYPE := LCP^.IDTYPE
2016:                  ELSE ERROR(108)
2017:              END;
2018:            INSYMBOL;
2019:          END
2020:        ELSE ERROR(2)
2021:      END (*POINTERTYPE*) ;
2022:
2023:    BEGIN (*TYP*)
2024:      PACKING := FALSE;
2025:      IF NOT (SY IN TYPEBEGSYS) THEN
2026:         BEGIN ERROR(10); SKIP(FSYS + TYPEBEGSYS) END;
2027:      IF SY IN TYPEBEGSYS THEN
2028:        BEGIN
2029:          IF SY IN SIMPTYPEBEGSYS THEN SIMPLETYPE(FSYS,FSP,FSIZE)
2030:          ELSE
2031:    (*^*)   IF SY = ARROW THEN POINTERTYPE
2032:          ELSE
2033:            BEGIN
2034:              IF SY = PACKEDSY THEN
2035:                BEGIN INSYMBOL; PACKING := TRUE;
2036:                  IF NOT (SY IN TYPEDELS) THEN
2037:                    BEGIN ERROR(10); SKIP(FSYS + TYPEDELS) END
2038:                END;
2039:    (*ARRAY*)   IF SY = ARRAYSY THEN
2040:                BEGIN INSYMBOL;
2041:                  IF SY = LBRACK THEN INSYMBOL ELSE ERROR(11);
2042:                  LSP1 := NIL;
2043:                  REPEAT
2044:                    IF PACKING THEN NEW(LSP,ARRAYS,TRUE,FALSE)
2045:                    ELSE NEW(LSP,ARRAYS,FALSE);
2046:                    WITH LSP^ DO
```

```
2047:                       BEGIN AELTYPE := LSP1; INXTYPE := NIL;
2048:                         IF PACKING THEN AISSTRNG := FALSE;
2049:                         AISPACKD := FALSE;  FORM := ARRAYS
2050:                       END;
2051:                     LSP1 := LSP;
2052:                     SIMPLETYPE(FSYS + [COMMA,RBRACK,OFSY],LSP2,LSIZE);
2053:                     LSP1^.SIZE := LSIZE;
2054:                     IF LSP2 <> NIL THEN
2055:                       IF LSP2^.FORM <= SUBRANGE THEN
2056:                         BEGIN
2057:                           IF LSP2 = REALPTR THEN
2058:                             BEGIN ERROR(109); LSP2 := NIL END
2059:                           ELSE
2060:                             IF LSP2 = INTPTR THEN
2061:                               BEGIN ERROR(149); LSP2 := NIL END;
2062:                           LSP^.INXTYPE := LSP2
2063:                         END
2064:                       ELSE BEGIN ERROR(113); LSP2 := NIL END;
2065:                     TEST := SY <> COMMA;
2066:                     IF NOT TEST THEN INSYMBOL
2067:                   UNTIL TEST;
2068:                   IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12);
2069:                   IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
2070:                   TYP(FSYS,LSP,LSIZE);
2071:                   IF LSP <> NIL THEN
2072:                     IF LSP^.FORM = FILES THEN ERROR(108);
2073:                   IF PACKABLE(LSP) THEN
2074:                     IF NUMBITS + NUMBITS <= BITSPERWD THEN
2075:                       WITH LSP1^ DO
2076:                         BEGIN AISPACKD := TRUE;
2077:                           ELSPERWD := BITSPERWD DIV NUMBITS;
2078:                           ELWIDTH := NUMBITS
2079:                         END;
2080:                   REPEAT
2081:                     WITH LSP1^ DO
2082:                       BEGIN LSP2 := AELTYPE; AELTYPE := LSP;
2083:                         IF INXTYPE <> NIL THEN
2084:                           BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
2085:                             IF AISPACKD THEN
2086:                               LSIZE := (LMAX-LMIN+ELSPERWD)
2087:                                                   DIV ELSPERWD
2088:                             ELSE
2089:                               LSIZE := LSIZE*(LMAX - LMIN + 1);
2090:                             IF LSIZE <= 0 THEN
2091:                               BEGIN ERROR(398); LSIZE := 1 END;
2092:                             SIZE := LSIZE
2093:                           END
2094:                       END;
2095:                     LSP := LSP1; LSP1 := LSP2
2096:                   UNTIL LSP1 = NIL
2097:                 END
2098:               ELSE
2099:   (*RECORD*)    IF SY = RECORDSY THEN
2100:                   BEGIN INSYMBOL;
2101:                     OLDTOP := TOP;
2102:                     IF TOP < DISPLIMIT THEN
2103:                       BEGIN TOP := TOP + 1;
2104:                         WITH DISPLAY[TOP] DO
2105:                           BEGIN FNAME := NIL; OCCUR := REC END
2106:                       END
2107:                     ELSE ERROR(250);
2108:                     DISPL := 0; NEXTBIT := 0;
2109:                     FIELDLIST(FSYS-[SEMICOLON]+[ENDSY],LSP1);
2110:                     DISPL := DISPL + ORD(NEXTBIT > 0);
2111:                     NEW(LSP,RECORDS);
2112:                     WITH LSP^ DO
```

```
2113:                       BEGIN FSTFLD := DISPLAY[TOP].FNAME;
2114:                         RECVAR := LSP1; SIZE := DISPL;
2115:                         FORM := RECORDS
2116:                       END;
2117:                     TOP := OLDTOP;
2118:                     IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
2119:                   END
2120:                 ELSE
2121:   (*SET*)         IF SY = SETSY THEN
2122:                     BEGIN INSYMBOL;
2123:                       IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
2124:                       SIMPLETYPE(FSYS,LSP1,LSIZE);
2125:                       IF LSP1 <> NIL THEN
2126:                         IF (LSP1^.FORM > SUBRANGE) OR
2127:                                 (LSP1 = INTPTR) OR (LSP1 = REALPTR) THEN
2128:                           BEGIN ERROR(115); LSP1 := NIL END
2129:                         ELSE
2130:                           IF LSP1 = REALPTR THEN
2131:                             BEGIN ERROR(114); LSP1 := NIL END;
2132:                       NEW(LSP,POWER);
2133:                       WITH LSP^ DO
2134:                         BEGIN ELSET := LSP1; FORM := POWER;
2135:                           IF LSP1 <> NIL THEN
2136:                             BEGIN GETBOUNDS(LSP1,LMIN,LMAX);
2137:                               SIZE := (LMAX + BITSPERWD) DIV BITSPERWD;
2138:                               IF SIZE > 255 THEN
2139:                                 BEGIN ERROR(169); SIZE := 1 END
2140:                             END
2141:                           ELSE SIZE := 0
2142:                         END
2143:                     END
2144:                   ELSE
2145:   (*FILE*)          IF SY = FILESY THEN
2146:                       BEGIN INSYMBOL; NEW(LSP,FILES);
2147:                         WITH LSP^ DO
2148:                           BEGIN FORM := FILES; FILTYPE := NIL END;
2149:                         IF SY = OFSY THEN
2150:                           BEGIN INSYMBOL; TYP(FSYS,LSP1,LSIZE) END
2151:                         ELSE LSP1 := NIL;
2152:                         LSP^.FILTYPE := LSP1;
2153:                         IF LSP1 <> NIL THEN
2154:                           LSP^.SIZE := FILESIZE + LSP1^.SIZE
2155:                         ELSE LSP^.SIZE := NILFILESIZE
2156:                       END;
2157:                 FSP := LSP
2158:               END;
2159:           IF NOT (SY IN FSYS) THEN
2160:             BEGIN ERROR(6); SKIP(FSYS) END
2161:         END
2162:       ELSE FSP := NIL;
2163:       IF FSP = NIL THEN FSIZE := 1 ELSE FSIZE := FSP^.SIZE
2164:     END (*TYP*) ;
2165:    (*    COPYRIGHT (C) 1978, REGENTS OF THE       *)
2166:   (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
2167:
2168:     PROCEDURE GENLDC(IVAL: INTEGER); FORWARD;
2169:
2170:     PROCEDURE GENBYTE(FBYTE: INTEGER);
2171:     BEGIN
2172:       CODEP^[IC] := CHR(FBYTE); IC := IC+1
2173:     END (*GENBYTE*) ;
2174:
2175:     PROCEDURE GENWORD(FWORD: INTEGER);
2176:     BEGIN
2177:       IF ODD(IC) THEN IC := IC + 1;
2178:       MOVELEFT(FWORD,CODEP^[IC],2);
```

```
2179:       IC := IC + 2
2180:     END (*GENWORD*) ;
2181:
2182:     PROCEDURE GENBIG(IVAL: INTEGER);
2183:       VAR LOWORDER: CHAR;
2184:     BEGIN
2185:       IF IVAL <= 127 THEN GENBYTE(IVAL)
2186:       ELSE
2187:         BEGIN MOVELEFT(IVAL,CODEP^[IC],2); LOWORDER := CODEP^[IC];
2188:           CODEP^[IC] := CHR(ORD(CODEP^[IC+1])+128);
2189:           CODEP^[IC+1] := LOWORDER; IC := IC+2
2190:         END
2191:     END (*GENBIG*) ;
2192:
2193:     PROCEDURE GEN0(FOP: OPRANGE);
2194:       VAR I: INTEGER;
2195:     BEGIN
2196:       GENBYTE(FOP+128);
2197:       IF FOP = 38(*LCA*) THEN
2198:         WITH GATTR.CVAL.VALP^ DO
2199:           BEGIN GENBYTE(SLGTH);
2200:             FOR I := 1 TO SLGTH DO GENBYTE(ORD(SVAL[I]))
2201:           END
2202:     END (*GEN0*) ;
2203:
2204:     PROCEDURE GEN1(FOP: OPRANGE; FP2: INTEGER);
2205:       LABEL 1;
2206:       VAR I,J: INTEGER;
2207:     BEGIN
2208:       GENBYTE(FOP+128);
2209:       IF FOP = 51(*LDC*) THEN
2210:         BEGIN
2211:           IF FP2 = 2 THEN I := REALSIZE
2212:           ELSE
2213:             BEGIN I := 8;
2214:               WHILE I > 0 DO
2215:                 IF GATTR.CVAL.VALP^.CSTVAL[I] <> 0 THEN GOTO 1
2216:                 ELSE I := I - 1;
2217:         1:  END;
2218:           GATTR.TYPTR^.SIZE := I;
2219:           IF I > 1 THEN
2220:             BEGIN GENBYTE(I);
2221:               FOR J := I DOWNTO 1 DO GENWORD(GATTR.CVAL.VALP^.CSTVAL[J])
2222:             END
2223:           ELSE
2224:             BEGIN IC := IC - 1;
2225:               IF I = 1 THEN GENLDC(GATTR.CVAL.VALP^.CSTVAL[1])
2226:             END
2227:         END
2228:       ELSE
2229:         IF FOP IN [30(*CSP*),32(*ADJ*),45(*RNP*),
2230:                    46(*CIP*),60(*LDM*),61(*STM*),
2231:                    65(*RBP*),66(*CBP*),78(*CLP*),
2232:                    42(*SAS*),79(*CGP*)] THEN GENBYTE(FP2)
2233:         ELSE
2234:           IF ((FOP = 74(*LDL*)) OR (FOP = 39(*LDO*)))
2235:               AND (FP2 <= 16) THEN
2236:             BEGIN IC := IC-1;
2237:               IF FOP = 39(*LDO*) THEN GENBYTE(231+FP2)
2238:               ELSE GENBYTE(215+FP2)
2239:             END
2240:           ELSE
2241:             IF (FOP = 35(*IND*)) AND (FP2 <= 7) THEN
2242:               BEGIN IC := IC-1; GENBYTE(248+FP2) END
2243:             ELSE GENBIG(FP2)
2244:     END (*GEN1*) ;
```

```
2245:
2246:    PROCEDURE GEN2(FOP: OPRANGE; FP1,FP2: INTEGER);
2247:    BEGIN
2248:      IF (FOP = 64(*IXP*)) OR (FOP = 77(*CXP*)) THEN
2249:        BEGIN GENBYTE(FOP+128); GENBYTE(FP1); GENBYTE(FP2);
2250:        END
2251:      ELSE
2252:        IF FOP IN [47(*EQU*),48(*GEQ*),49(*GRT*),
2253:                   52(*LEQ*),53(*LES*),55(*NEQ*)] THEN
2254:          IF FP1 = 0 THEN GEN0(FOP+20)
2255:          ELSE
2256:            BEGIN GEN1(FOP,FP1+FP1);
2257:              IF FP1 > 4 THEN GENBIG(FP2)
2258:            END
2259:        ELSE
2260:          BEGIN (*LDA,LOD,STR*)
2261:            IF FP1 = 0 THEN GEN1(FOP+20,FP2)
2262:            ELSE
2263:              BEGIN
2264:                GENBYTE(FOP+128); GENBYTE(FP1); GENBIG(FP2)
2265:              END
2266:          END;
2267:    END (*GEN2*) ;
2268:
2269:    PROCEDURE GENLDC;
2270:    BEGIN
2271:      IF (IVAL >= 0) AND (IVAL <= 127) THEN GENBYTE(IVAL)
2272:      ELSE
2273:        BEGIN GENBYTE(51(*LDC*)+148);
2274:          MOVELEFT(IVAL,CODEP^[IC],2);
2275:          IC := IC+2
2276:        END
2277:    END (*GENLDC*) ;
2278:
2279:    PROCEDURE GENJMP(FOP: OPRANGE; FLBP: LBP);
2280:      VAR DISP: INTEGER;
2281:    BEGIN
2282:      WITH FLBP^ DO
2283:        IF DEFINED THEN
2284:          BEGIN
2285:            GENBYTE(FOP+128);
2286:            DISP := OCCURIC-IC-1;
2287:            IF (DISP >= 0) AND (DISP <= 127) THEN GENBYTE(DISP)
2288:            ELSE
2289:              BEGIN
2290:                IF JTABINX = 0 THEN
2291:                  BEGIN JTABINX := NEXTJTAB;
2292:                    IF NEXTJTAB = MAXJTAB THEN ERROR(253)
2293:                    ELSE NEXTJTAB := NEXTJTAB + 1;
2294:                    JTAB[JTABINX] := OCCURIC
2295:                  END;
2296:                DISP := -JTABINX;
2297:                GENBYTE(248-JTABINX-JTABINX)
2298:              END;
2299:          END
2300:        ELSE
2301:          BEGIN MOVELEFT(REFLIST,CODEP^[IC],2);
2302:            IF FOP = 57(*UJP*) THEN DISP := IC + 4096
2303:            ELSE DISP := IC;
2304:            REFLIST := DISP; IC := IC+2
2305:          END;
2306:    END (*GENJMP*) ;
2307:
2308:    PROCEDURE LOAD; FORWARD;
2309:
2310:    PROCEDURE GENFJP(FLBP: LBP);
```

```
2311:    BEGIN LOAD;
2312:      IF GATTR.TYPTR <> BOOLPTR THEN ERROR(135);
2313:      GENJMP(33(*FJP*),FLBP)
2314:    END (*GENFJP*) ;
2315:
2316:    PROCEDURE GENLABEL(VAR FLBP: LBP);
2317:    BEGIN NEW(FLBP);
2318:      WITH FLBP^ DO
2319:        BEGIN DEFINED := FALSE; REFLIST := MAXADDR END
2320:    END (*GENLABEL*) ;
2321:
2322:    PROCEDURE PUTLABEL(FLBP: LBP);
2323:      VAR LREF: INTEGER; LOP: OPRANGE;
2324:    BEGIN
2325:      WITH FLBP^ DO
2326:        BEGIN LREF := REFLIST;
2327:          DEFINED := TRUE; OCCURIC := IC; JTABINX := 0;
2328:          WHILE LREF < MAXADDR DO
2329:            BEGIN
2330:              IF LREF >= 4096 THEN
2331:                BEGIN LREF := LREF - 4096; LOP := 57(*UJP*) END
2332:              ELSE LOP := 33(*FJP*);
2333:              IC := LREF;
2334:              MOVELEFT(CODEP^[IC],LREF,2);
2335:              GENJMP(LOP,FLBP)
2336:            END;
2337:          IC := OCCURIC
2338:        END
2339:    END (*PUTLABEL*) ;
2340:
2341:    PROCEDURE LOAD;
2342:    BEGIN
2343:      WITH GATTR DO
2344:        IF TYPTR <> NIL THEN
2345:          BEGIN
2346:            CASE KIND OF
2347:              CST:   IF (TYPTR^.FORM = SCALAR) AND (TYPTR <> REALPTR) THEN
2348:                       GENLDC(CVAL.IVAL)
2349:                     ELSE
2350:                       IF TYPTR = NILPTR THEN GEN0(31(*LDCN*))
2351:                       ELSE
2352:                         IF TYPTR = REALPTR THEN GEN1(51(*LDC*),2)
2353:                         ELSE GEN1(51(*LDC*),5);
2354:              VARBL: CASE ACCESS OF
2355:                       DRCT:   IF VLEVEL = 1 THEN GEN1(39(*LDO*),DPLMT)
2356:                               ELSE GEN2(54(*LOD*),LEVEL-VLEVEL,DPLMT);
2357:                       INDRCT: GEN1(35(*IND*),IDPLMT);
2358:                       PACKD:  GEN0(58(*LDP*));
2359:                       MULTI:  GEN1(60(*LDM*),TYPTR^.SIZE);
2360:                       BYTE:   GEN0(62(*LDB*))
2361:                     END;
2362:              EXPR:
2363:            END;
2364:            IF (TYPTR^.FORM = POWER) AND (KIND <> EXPR) THEN
2365:              GENLDC(TYPTR^.SIZE);
2366:            KIND := EXPR
2367:          END
2368:    END (*LOAD*) ;
2369:
2370:    PROCEDURE STORE(VAR FATTR: ATTR);
2371:    BEGIN
2372:      WITH FATTR DO
2373:        IF TYPTR <> NIL THEN
2374:          CASE ACCESS OF
2375:            DRCT:   IF VLEVEL = 1 THEN GEN1(43(*SRO*),DPLMT)
2376:                    ELSE GEN2(56(*STR*),LEVEL-VLEVEL,DPLMT);
```

```
2377:           INDRCT: IF IDPLMT <> 0 THEN ERROR(400)
2378:                   ELSE GEN0(26(*STO*));
2379:           PACKD:  GEN0(59(*STP*));
2380:           MULTI:  GEN1(61(*STM*),TYPTR^.SIZE);
2381:           BYTE:   GEN0(63(*STB*))
2382:         END
2383:    END (*STORE*) ;
2384:
2385:    PROCEDURE LOADADDRESS;
2386:    BEGIN
2387:      WITH GATTR DO
2388:        IF TYPTR <> NIL THEN
2389:          BEGIN
2390:            CASE KIND OF
2391:              CST:   IF STRGTYPE(TYPTR) THEN GEN0(38(*LCA*))
2392:                     ELSE ERROR(400);
2393:              VARBL: CASE ACCESS OF
2394:                       DRCT:   IF VLEVEL = 1 THEN GEN1(37(*LAO*),DPLMT)
2395:                               ELSE GEN2(50(*LDA*),LEVEL-VLEVEL,DPLMT);
2396:                       INDRCT: IF IDPLMT <> 0 THEN GEN1(34(*INC*),IDPLMT+IDPLMT);
2397:                       PACKD:  ERROR(103)
2398:                     END
2399:            END;
2400:            KIND := VARBL; ACCESS := INDRCT; IDPLMT := 0
2401:          END
2402:    END (*LOADADDRESS*) ;
2403:
2404:    PROCEDURE WRITECODE(FORCEBUF: BOOLEAN);
2405:      VAR CODEINX,LIC,I: INTEGER;
2406:    BEGIN CODEINX := 0; LIC := IC;
2407:      REPEAT
2408:        I := 512-CURBYTE;
2409:        IF I > LIC THEN I := LIC;
2410:        MOVELEFT(CODEP^[CODEINX],DISKBUF[CURBYTE],I);
2411:        CODEINX := CODEINX+I;
2412:        CURBYTE := CURBYTE+I;
2413:        IF (CURBYTE = 512) OR FORCEBUF THEN
2414:          BEGIN
2415:            IF USERINFO.ERRNUM = 0 THEN
2416:              IF BLOCKWRITE(USERINFO.WORKCODE^,DISKBUF,1,CURBLK) <> 1 THEN
2417:                ERROR(402);
2418:            CURBLK := CURBLK+1; CURBYTE := 0
2419:          END;
2420:        LIC := LIC-I
2421:      UNTIL LIC = 0;
2422:    END (*WRITECODE*) ;
2423:
2424:    PROCEDURE FINISHSEG;
2425:      VAR I: INTEGER;
2426:    BEGIN IC := 0;
2427:      FOR I := NEXTPROC-1 DOWNTO 1 DO GENWORD(SEGINX+IC-PROCTABLE[I]);
2428:      GENBYTE(SEG); GENBYTE(NEXTPROC-1);
2429:      SEGTABLE[SEG].CODELENG := SEGINX+IC;
2430:      WRITECODE(TRUE); SEGINX := 0; CODEINSEG := FALSE
2431:    END (*FINISHSEG*) ;
2432:
2433:    PROCEDURE EXPRESSION(FSYS: SETOFSYS); FORWARD;
2434:
2435:    PROCEDURE SELECTOR(FSYS: SETOFSYS; FCP: CTP);
2436:      VAR LATTR: ATTR; LCP: CTP; LMIN,LMAX: INTEGER;
2437:    BEGIN
2438:      WITH FCP^, GATTR DO
2439:        BEGIN TYPTR := IDTYPE; KIND := VARBL;
2440:          CASE KLASS OF
2441:            VARS:
2442:              IF VKIND = ACTUAL THEN
```

```
2443:               BEGIN ACCESS := DRCT; VLEVEL := VLEV;
2444:                 DPLMT := VADDR
2445:               END
2446:             ELSE
2447:               BEGIN
2448:                 IF VLEV = 1 THEN GEN1(39(*LDO*),VADDR)
2449:                 ELSE GEN2(54(*LOD*),LEVEL-VLEV,VADDR);
2450:                 ACCESS := INDRCT; IDPLMT := 0
2451:               END;
2452:           FIELD:
2453:             WITH DISPLAY[DISX] DO
2454:              BEGIN
2455:               IF OCCUR = CREC THEN
2456:                 BEGIN ACCESS := DRCT; VLEVEL := CLEV;
2457:                   DPLMT := CDSPL + FLDADDR
2458:                 END
2459:               ELSE
2460:                 BEGIN
2461:                   IF LEVEL = 1 THEN GEN1(39(*LDO*),VDSPL)
2462:                   ELSE GEN2(54(*LOD*),0,VDSPL);
2463:                   ACCESS := INDRCT; IDPLMT := FLDADDR
2464:                 END;
2465:               IF FISPACKD THEN
2466:                 BEGIN LOADADDRESS;
2467:                   IF ((FLDRBIT = 0) OR (FLDRBIT = 8))
2468:                         AND (FLDWIDTH = 8) THEN
2469:                     BEGIN ACCESS := BYTE;
2470:                       IF FLDRBIT = 8 THEN GEN1(34(*INC*),1)
2471:                     END
2472:                   ELSE
2473:                     BEGIN ACCESS := PACKD;
2474:                       GENLDC(FLDWIDTH); GENLDC(FLDRBIT)
2475:                     END
2476:                 END
2477:              END;
2478:           FUNC:
2479:             IF PFDECKIND <> DECLARED THEN ERROR(150)
2480:             ELSE
2481:               IF NOT INSCOPE THEN ERROR(103)
2482:               ELSE
2483:                   BEGIN ACCESS := DRCT; VLEVEL := PFLEV + 1;
2484:                     DPLMT := LCAFTERMARKSTACK
2485:                   END
2486:         END (*CASE*);
2487:         IF TYPTR <> NIL THEN
2488:           IF (TYPTR^.FORM <= POWER) AND
2489:              (TYPTR^.SIZE > PTRSIZE) THEN
2490:             BEGIN LOADADDRESS; ACCESS := MULTI END
2491:       END (*WITH*);
2492:     IF NOT (SY IN SELECTSYS + FSYS) THEN
2493:       BEGIN ERROR(59); SKIP(SELECTSYS + FSYS) END;
2494:     WHILE SY IN SELECTSYS DO
2495:       BEGIN
2496:   (*[*) IF SY = LBRACK THEN
2497:           BEGIN
2498:             REPEAT LATTR := GATTR;
2499:               WITH LATTR DO
2500:                 IF TYPTR <> NIL THEN
2501:                   IF TYPTR^.FORM <> ARRAYS THEN
2502:                     BEGIN ERROR(138); TYPTR := NIL END;
2503:               LOADADDRESS;
2504:               INSYMBOL; EXPRESSION(FSYS + [COMMA,RBRACK]);
2505:               LOAD;
2506:               IF GATTR.TYPTR <> NIL THEN
2507:                 IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(113);
2508:               IF LATTR.TYPTR <> NIL THEN
```

```
2509:                 WITH LATTR.TYPTR^ DO
2510:                   BEGIN
2511:                     IF COMPTYPES(INXTYPE,GATTR.TYPTR) THEN
2512:                       BEGIN
2513:                         IF (INXTYPE <> NIL) AND
2514:                             NOT STRGTYPE(LATTR.TYPTR) THEN
2515:                           BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
2516:                             IF RANGECHECK THEN
2517:                               BEGIN GENLDC(LMIN); GENLDC(LMAX);
2518:                                 GEN0(8(*CHK*))
2519:                               END;
2520:                             IF LMIN <> 0 THEN
2521:                               BEGIN GENLDC(ABS(LMIN));
2522:                                 IF LMIN > 0 THEN GEN0(21(*SBI*))
2523:                                 ELSE GEN0(2(*ADI*))
2524:                               END
2525:                           END
2526:                       END
2527:                     ELSE ERROR(139);
2528:                     WITH GATTR DO
2529:                       BEGIN TYPTR := AELTYPE; KIND := VARBL;
2530:                         ACCESS := INDRCT; IDPLMT := 0;
2531:                         IF TYPTR <> NIL THEN
2532:                           IF AISPACKD THEN
2533:                             IF ELWIDTH = 8 THEN
2534:                               BEGIN ACCESS := BYTE;
2535:                                 IF STRGTYPE(LATTR.TYPTR) AND RANGECHECK THEN
2536:                                   GEN0(27(*IXS*))
2537:                                 ELSE GEN0(2(*ADI*))
2538:                               END
2539:                             ELSE
2540:                               BEGIN ACCESS := PACKD;
2541:                                 GEN2(64(*IXP*),ELSPERWD,ELWIDTH)
2542:                               END
2543:                           ELSE
2544:                             BEGIN GEN1(36(*IXA*),TYPTR^.SIZE);
2545:                               IF (TYPTR^.FORM <= POWER) AND
2546:                                  (TYPTR^.SIZE > PTRSIZE) THEN
2547:                                 ACCESS := MULTI
2548:                               END
2549:                       END
2550:                   END
2551:               UNTIL SY <> COMMA;
2552:               IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
2553:             END (*IF SY = LBRACK*)
2554:           ELSE
2555:     (*.*)   IF SY = PERIOD THEN
2556:               BEGIN
2557:                 WITH GATTR DO
2558:                   BEGIN
2559:                     IF TYPTR <> NIL THEN
2560:                       IF TYPTR^.FORM <> RECORDS THEN
2561:                         BEGIN ERROR(140); TYPTR := NIL END;
2562:                     INSYMBOL;
2563:                     IF SY = IDENT THEN
2564:                       BEGIN
2565:                         IF TYPTR <> NIL THEN
2566:                           BEGIN SEARCHSECTION(TYPTR^.FSTFLD,LCP);
2567:                             IF LCP = NIL THEN
2568:                               BEGIN ERROR(152); TYPTR := NIL END
2569:                             ELSE
2570:                               WITH LCP^ DO
2571:                                 BEGIN TYPTR := IDTYPE;
2572:                                   CASE ACCESS OF
2573:                                     DRCT:   DPLMT := DPLMT + FLDADDR;
2574:                                     INDRCT: IDPLMT := IDPLMT + FLDADDR;
```

```
2575:                                   MULTI,BYTE,
2576:                                   PACKD:  ERROR(400)
2577:                                 END (*CASE ACCESS*);
2578:                                 IF FISPACKD THEN
2579:                                   BEGIN LOADADDRESS;
2580:                                     IF ((FLDRBIT = 0) OR (FLDRBIT = 8))
2581:                                         AND (FLDWIDTH = 8) THEN
2582:                                       BEGIN ACCESS := BYTE;
2583:                                         IF FLDRBIT = 8 THEN GEN1(34(*INC*),1)
2584:                                       END
2585:                                     ELSE
2586:                                       BEGIN ACCESS := PACKD;
2587:                                         GENLDC(FLDWIDTH); GENLDC(FLDRBIT)
2588:                                       END
2589:                                   END;
2590:                                 IF TYPTR <> NIL THEN
2591:                                   IF (TYPTR^.FORM <= POWER) AND
2592:                                      (TYPTR^.SIZE > PTRSIZE) THEN
2593:                                     BEGIN LOADADDRESS; ACCESS := MULTI END
2594:                               END
2595:                           END;
2596:                         INSYMBOL
2597:                       END (*SY = IDENT*)
2598:                     ELSE ERROR(2)
2599:                   END (*WITH GATTR*)
2600:               END (*IF SY = PERIOD*)
2601:             ELSE
2602:    (*^*)      BEGIN
2603:                 IF GATTR.TYPTR <> NIL THEN
2604:                   WITH GATTR,TYPTR^ DO
2605:                     IF (FORM = POINTER) OR (FORM = FILES) THEN
2606:                       BEGIN LOAD; KIND := VARBL;
2607:                         ACCESS := INDRCT; IDPLMT := 0;
2608:                         IF FORM = POINTER THEN TYPTR := ELTYPE
2609:                         ELSE
2610:                           BEGIN TYPTR := FILTYPE;
2611:                             IF TYPTR = NIL THEN ERROR(399)
2612:                           END;
2613:                         IF TYPTR <> NIL THEN
2614:                           IF (TYPTR^.FORM <= POWER) AND
2615:                              (TYPTR^.SIZE > PTRSIZE) THEN
2616:                                   ACCESS := MULTI
2617:                       END
2618:                     ELSE ERROR(141);
2619:                 INSYMBOL
2620:               END;
2621:           IF NOT (SY IN FSYS + SELECTSYS) THEN
2622:             BEGIN ERROR(6); SKIP(FSYS + SELECTSYS) END
2623:         END (*WHILE*)
2624:     END (*SELECTOR*) ;
2625:    (*    COPYRIGHT (C) 1978, REGENTS OF THE       *)
2626:    (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO      *)
2627:
2628:     PROCEDURE CALL(FSYS: SETOFSYS; FCP: CTP);
2629:       VAR LKEY: 1..40; WASLPARENT: BOOLEAN;
2630:
2631:       PROCEDURE VARIABLE(FSYS: SETOFSYS);
2632:         VAR LCP: CTP;
2633:       BEGIN
2634:         IF SY = IDENT THEN
2635:           BEGIN SEARCHID([FIELD,VARS],LCP); INSYMBOL END
2636:         ELSE BEGIN ERROR(2); LCP := UVARPTR END;
2637:         SELECTOR(FSYS,LCP)
2638:       END (*VARIABLE*) ;
2639:
2640:       PROCEDURE STRGVAR(FSYS: SETOFSYS; MUSTBEVAR: BOOLEAN);
```

```
2641:      BEGIN EXPRESSION(FSYS);
2642:        WITH GATTR DO
2643:          IF ((KIND = CST) AND (TYPTR = CHARPTR))
2644:              OR STRGTYPE(TYPTR) THEN
2645:            IF KIND = VARBL THEN LOADADDRESS
2646:            ELSE
2647:              BEGIN
2648:                IF MUSTBEVAR THEN ERROR(154);
2649:                IF KIND = CST THEN
2650:                  BEGIN
2651:                    IF TYPTR = CHARPTR THEN
2652:                      BEGIN
2653:                        WITH SCONST^ DO
2654:                          BEGIN CCLASS := STRG; SLGTH := 1;
2655:                            SVAL[1] := CHR(CVAL.IVAL)
2656:                          END;
2657:                        CVAL.VALP := SCONST;
2658:                        NEW(TYPTR,ARRAYS,TRUE,TRUE);
2659:                        TYPTR^ := STRGPTR^;
2660:                        TYPTR^.MAXLENG := 1
2661:                      END;
2662:                    LOADADDRESS
2663:                  END
2664:              END
2665:          ELSE
2666:            BEGIN
2667:              IF GATTR.TYPTR <> NIL THEN ERROR(125);
2668:              GATTR.TYPTR := STRGPTR
2669:            END
2670:      END (*STRGVAR*) ;
2671:
2672:      PROCEDURE NEWSTMT;
2673:        LABEL 1;
2674:        VAR LSP,LSP1: STP; VARTS,LMIN,LMAX: INTEGER;
2675:            LSIZE,LSZ: ADDRRANGE; LVAL: VALU;
2676:      BEGIN VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2677:        LSP := NIL; VARTS := 0; LSIZE := 0;
2678:        IF GATTR.TYPTR <> NIL THEN
2679:          WITH GATTR.TYPTR^ DO
2680:            IF FORM = POINTER THEN
2681:              BEGIN
2682:                IF ELTYPE <> NIL THEN
2683:                  WITH ELTYPE^ DO
2684:                    BEGIN LSIZE := SIZE;
2685:                      IF FORM = RECORDS THEN LSP := RECVAR
2686:                    END
2687:              END
2688:            ELSE ERROR(116);
2689:        WHILE SY = COMMA DO
2690:          BEGIN INSYMBOL;
2691:            CONSTANT(FSYS + [COMMA,RPARENT],LSP1,LVAL);
2692:            VARTS := VARTS + 1;
2693:            IF LSP = NIL THEN ERROR(158)
2694:            ELSE
2695:              IF LSP^.FORM <> TAGFLD THEN ERROR(162)
2696:              ELSE
2697:                IF LSP^.TAGFIELDP <> NIL THEN
2698:                  IF STRGTYPE(LSP1) OR (LSP1 = REALPTR) THEN ERROR(159)
2699:                  ELSE
2700:                    IF COMPTYPES(LSP^.TAGFIELDP^.IDTYPE,LSP1) THEN
2701:                      BEGIN
2702:                        LSP1 := LSP^.FSTVAR;
2703:                        WHILE LSP1 <> NIL DO
2704:                          WITH LSP1^ DO
2705:                            IF VARVAL.IVAL = LVAL.IVAL THEN
2706:                              BEGIN LSIZE := SIZE; LSP := SUBVAR;
```

```
2707:                               GOTO 1
2708:                             END
2709:                           ELSE LSP1 := NXTVAR;
2710:                       LSIZE := LSP^.SIZE; LSP := NIL;
2711:                     END
2712:                   ELSE ERROR(116);
2713:     1:  END (*WHILE*) ;
2714:       GENLDC(LSIZE);
2715:       GEN1(30(*CSP*),1(*NEW*))
2716:     END (*NEWSTMT*) ;
2717:
2718:     PROCEDURE MOVE;
2719:     BEGIN VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2720:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2721:       IF LKEY = 27 THEN
2722:         BEGIN EXPRESSION(FSYS + [COMMA]); LOAD END
2723:       ELSE
2724:         BEGIN VARIABLE(FSYS + [COMMA]); LOADADDRESS END;
2725:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2726:       EXPRESSION(FSYS + [RPARENT]); LOAD;
2727:       IF LKEY = 27 THEN GEN1(30(*CSP*),10(*FLC*))
2728:       ELSE
2729:         IF LKEY = 21 THEN GEN1(30(*CSP*),2(*MVL*))
2730:         ELSE GEN1(30(*CSP*),3(*MVR*))
2731:     END (*MOVE*) ;
2732:
2733:     PROCEDURE EXIT;
2734:       VAR LCP: CTP;
2735:     BEGIN
2736:       IF SY = IDENT THEN
2737:         BEGIN SEARCHID([PROC,FUNC],LCP); INSYMBOL END
2738:       ELSE
2739:         IF (SY = PROGSY) THEN
2740:           BEGIN LCP := OUTERBLOCK; INSYMBOL END
2741:         ELSE LCP := NIL;
2742:       IF LCP <> NIL THEN
2743:         IF LCP^.PFDECKIND = DECLARED THEN
2744:           BEGIN GENLDC(LCP^.PFSEG); GENLDC(LCP^.PFNAME) END
2745:         ELSE ERROR(125)
2746:       ELSE ERROR(125);
2747:       GEN1(30(*CSP*),4(*XIT*))
2748:     END (*EXIT*) ;
2749:
2750:     PROCEDURE UNITIO;
2751:     BEGIN
2752:       IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2753:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2754:       VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2755:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2756:       EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2757:       IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2758:       IF SY = COMMA THEN
2759:         BEGIN INSYMBOL;
2760:           IF SY = COMMA THEN GENLDC(0)
2761:           ELSE
2762:             BEGIN
2763:               EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2764:               IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2765:             END
2766:         END
2767:       ELSE GENLDC(0);
2768:       IF SY = COMMA THEN
2769:         BEGIN INSYMBOL;
2770:           EXPRESSION(FSYS + [RPARENT]); LOAD;
2771:           IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2772:         END
```

```
2773:         ELSE GENLDC(0);
2774:         IF LKEY = 13 THEN GEN1(30(*CSP*),5(*URD*))
2775:         ELSE GEN1(30(*CSP*),6(*UWT*))
2776:       END (*UNITIO*);
2777:
2778:       PROCEDURE CONCAT;
2779:         VAR LLC: ADDRRANGE; TEMPLGTH: INTEGER;
2780:       BEGIN TEMPLGTH := 0;
2781:         LLC := LC; LC := LC + (STRGLGTH DIV CHRSPERWD) + 1;
2782:         GENLDC(0); GEN2(56(*STR*),0,LLC);
2783:         GEN2(50(*LDA*),0,LLC);
2784:         REPEAT
2785:           STRGVAR(FSYS + [COMMA,RPARENT],FALSE);
2786:           TEMPLGTH := TEMPLGTH + GATTR.TYPTR^.MAXLENG;
2787:           IF TEMPLGTH < STRGLGTH THEN GENLDC(TEMPLGTH)
2788:           ELSE GENLDC(STRGLGTH);
2789:           GEN2(77(*CXP*),0(*SYS*),23(*SCONCAT*));
2790:           GEN2(50(*LDA*),0,LLC);
2791:           TEST := SY <> COMMA;
2792:           IF NOT TEST THEN INSYMBOL
2793:         UNTIL TEST;
2794:         IF TEMPLGTH < STRGLGTH THEN
2795:           LC := LLC + (TEMPLGTH DIV CHRSPERWD) + 1
2796:         ELSE TEMPLGTH := STRGLGTH;
2797:         IF LC > LCMAX THEN LCMAX := LC;
2798:         LC := LLC;
2799:         WITH GATTR DO
2800:           BEGIN NEW(TYPTR,ARRAYS,TRUE,TRUE);
2801:             TYPTR^ := STRGPTR^;
2802:             TYPTR^.MAXLENG := TEMPLGTH
2803:           END
2804:       END (*CONCAT*) ;
2805:
2806:       PROCEDURE COPYDELETE;
2807:         VAR LLC: ADDRRANGE; LSP: STP;
2808:       BEGIN
2809:         IF LKEY = 19 THEN
2810:           BEGIN LLC := LC;
2811:             LC := LC + (STRGLGTH DIV CHRSPERWD) + 1;
2812:           END;
2813:         IF LKEY <> 43 THEN
2814:           BEGIN
2815:             STRGVAR(FSYS + [COMMA], LKEY = 18);
2816:             IF LKEY = 19 THEN
2817:               BEGIN LSP := GATTR.TYPTR;
2818:                 GEN2(50(*LDA*),0,LLC)
2819:               END;
2820:             IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2821:           END;
2822:         EXPRESSION(FSYS + [COMMA]); LOAD;
2823:         IF GATTR.TYPTR <> NIL THEN
2824:           IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2825:         IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2826:         EXPRESSION(FSYS + [RPARENT]); LOAD;
2827:         IF GATTR.TYPTR <> NIL THEN
2828:           IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2829:         IF LKEY = 19 THEN
2830:           BEGIN
2831:             GEN2(77(*CXP*),0(*SYS*),25(*SCOPY*));
2832:             GEN2(50(*LDA*),0,LLC);
2833:             IF LSP^.MAXLENG < STRGLGTH THEN
2834:               LC := LLC + (LSP^.MAXLENG DIV CHRSPERWD) + 1;
2835:             IF LC > LCMAX THEN LCMAX := LC;
2836:             LC := LLC; GATTR.TYPTR := LSP
2837:           END
2838:         ELSE
```

```
2839:          IF LKEY = 43 THEN
2840:            GEN2(77(*CXP*),0(*SYS*),29(*GOTOXY*))
2841:          ELSE GEN2(77(*CXP*),0(*SYS*),26(*SDELETE*))
2842:      END (*COPYDELETE*) ;
2843:
2844:      PROCEDURE CLOSE;
2845:      BEGIN
2846:        VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2847:        IF GATTR.TYPTR <> NIL THEN
2848:          IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125);
2849:        IF SY = COMMA THEN
2850:          BEGIN INSYMBOL;
2851:            IF SY = IDENT THEN
2852:             BEGIN
2853:              IF ID = 'NORMAL  ' THEN GENLDC(0)
2854:              ELSE
2855:                IF ID = 'LOCK    ' THEN GENLDC(1)
2856:                ELSE
2857:                  IF ID = 'PURGE   ' THEN GENLDC(2)
2858:                  ELSE
2859:                    IF ID = 'CRUNCH  ' THEN GENLDC(3)
2860:                    ELSE ERROR(2);
2861:              INSYMBOL
2862:             END
2863:            ELSE ERROR(2)
2864:          END
2865:        ELSE GENLDC(0);
2866:        GEN2(77(*CXP*),0(*SYS*),6(*FCLOSE*));
2867:        IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2868:      END (*CLOSE*) ;
2869:
2870:      PROCEDURE GETPUTETC;
2871:      BEGIN
2872:        VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2873:        IF GATTR.TYPTR <> NIL THEN
2874:          IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
2875:          ELSE
2876:            IF GATTR.TYPTR^.FILTYPE = NIL THEN ERROR(399);
2877:        CASE LKEY OF
2878:          32:  BEGIN
2879:                 IF SY = COMMA THEN
2880:                   BEGIN
2881:                     INSYMBOL; EXPRESSION(FSYS + [RPARENT]); LOAD;
2882:                     IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2883:                   END
2884:                 ELSE ERROR(125);
2885:                 GEN2(77(*CXP*),0(*SYS*),9(*FSEEK*))
2886:               END;
2887:          34:  GEN2(77(*CXP*),0(*SYS*),7(*FGET*));
2888:          35:  GEN2(77(*CXP*),0(*SYS*),8(*FPUT*));
2889:          40:  BEGIN
2890:                 IF GATTR.TYPTR <> NIL THEN
2891:                   IF GATTR.TYPTR^.FILTYPE <> CHARPTR THEN ERROR(399);
2892:                 GENLDC(12); GENLDC(0);
2893:                 GEN2(77(*CXP*),0(*SYS*),17(*WRC*))
2894:               END
2895:        END (*CASE*) ;
2896:        IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2897:      END (*GETPUTETC*) ;
2898:
2899:      PROCEDURE SCAN;
2900:      BEGIN
2901:        IF GATTR.TYPTR <> NIL THEN
2902:          IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2903:        IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2904:        IF SY = RELOP THEN
```

```
2905:           BEGIN
2906:             IF OP = EQOP THEN GENLDC(0)
2907:             ELSE
2908:               IF OP = NEOP THEN GENLDC(1)
2909:               ELSE ERROR(125);
2910:             INSYMBOL
2911:           END
2912:         ELSE ERROR(125);
2913:         EXPRESSION(FSYS + [COMMA]); LOAD;
2914:         IF GATTR.TYPTR <> NIL THEN
2915:           IF GATTR.TYPTR <> CHARPTR THEN ERROR(125);
2916:         IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2917:         VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2918:         IF SY = COMMA THEN
2919:           BEGIN INSYMBOL;
2920:             EXPRESSION(FSYS + [RPARENT]); LOAD
2921:           END
2922:         ELSE GENLDC(0);
2923:         GEN1(30(*CSP*),11(*SCN*));
2924:         GATTR.TYPTR := INTPTR
2925:       END (*SCAN*) ;
2926:
2927:       PROCEDURE BLOCKIO;
2928:       BEGIN
2929:         VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2930:         IF GATTR.TYPTR <> NIL THEN
2931:           IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
2932:           ELSE
2933:             IF GATTR.TYPTR^.FILTYPE <> NIL THEN ERROR(399);
2934:         IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2935:         VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2936:         IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2937:         EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2938:         IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2939:         IF SY = COMMA THEN
2940:           BEGIN INSYMBOL;
2941:             EXPRESSION(FSYS + [RPARENT]); LOAD;
2942:             IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2943:           END
2944:         ELSE GENLDC(-1);
2945:         IF LKEY = 37 THEN GENLDC(1) ELSE GENLDC(0);
2946:         GENLDC(0); GENLDC(0);
2947:         GEN2(77(*CXP*),0(*SYS*),28(*BLOCKIO*));
2948:         IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
2949:         GATTR.TYPTR := INTPTR
2950:       END (*BLOCKIO*) ;
2951:
2952:       PROCEDURE DRAWSTUFF;
2953:         VAR I,N: INTEGER;
2954:       BEGIN
2955:         VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2956:         IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2957:         VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2958:         IF LKEY = 42 THEN N := 6
2959:         ELSE N := 5;
2960:         FOR I := 0 TO N DO
2961:           BEGIN
2962:             IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2963:             EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2964:             IF GATTR.TYPTR <> NIL THEN
2965:               IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2966:           END;
2967:         IF LKEY = 42 THEN N := 13
2968:         ELSE N := 12;
2969:         GEN1(30(*CSP*),N)
2970:       END (*DRAWSTUFF*) ;
```

```
2971:
2972:       PROCEDURE SIZEOF;
2973:         VAR LCP: CTP;
2974:       BEGIN
2975:         IF SY = IDENT THEN
2976:           BEGIN SEARCHID([TYPES,VARS,FIELD],LCP); INSYMBOL;
2977:             IF LCP^.IDTYPE <> NIL THEN
2978:               GENLDC(LCP^.IDTYPE^.SIZE*CHRSPERWD)
2979:           END;
2980:         GATTR.TYPTR := INTPTR
2981:       END (*SIZEOF*) ;
2982:
2983:
2984:       PROCEDURE LOADIDADDR(FCP: CTP);
2985:       BEGIN
2986:           WITH FCP^ DO
2987:             IF VKIND = ACTUAL THEN
2988:               IF VLEV = 1 THEN GEN1(37(*LAO*),VADDR)
2989:               ELSE GEN2(50(*LDA*),LEVEL-VLEV,VADDR)
2990:             ELSE
2991:               IF VLEV = 1 THEN GEN1(39(*LDO*),VADDR)
2992:               ELSE GEN2(54(*LOD*),LEVEL-VLEV,VADDR)
2993:       END (*LOADIDADDR*) ;
2994:
2995:       PROCEDURE READ;
2996:         VAR FILEPTR,LCP: CTP;
2997:       BEGIN FILEPTR := INPUTPTR;
2998:         IF (SY = IDENT) AND WASLPARENT THEN
2999:           BEGIN SEARCHID([FIELD,VARS],LCP);
3000:             IF LCP^.IDTYPE <> NIL THEN
3001:               IF LCP^.IDTYPE^.FORM = FILES THEN
3002:                 IF LCP^.IDTYPE^.FILTYPE = CHARPTR THEN
3003:                   BEGIN INSYMBOL; FILEPTR := LCP;
3004:                     IF NOT (SY IN [COMMA,RPARENT]) THEN ERROR(20);
3005:                     IF SY = COMMA THEN INSYMBOL
3006:                   END
3007:           END
3008:         ELSE
3009:           IF WASLPARENT THEN ERROR(2);
3010:         IF WASLPARENT AND (SY <> RPARENT) THEN
3011:           BEGIN
3012:             REPEAT LOADIDADDR(FILEPTR);
3013:               VARIABLE(FSYS + [COMMA,RPARENT]);
3014:               IF GATTR.ACCESS = BYTE THEN ERROR(103);
3015:               LOADADDRESS;
3016:               IF GATTR.TYPTR <> NIL THEN
3017:                 IF COMPTYPES(INTPTR,GATTR.TYPTR) THEN
3018:                   GEN2(77(*CXP*),0(*SYS*),12(*FRDI*))
3019:                 ELSE
3020:                   IF COMPTYPES(REALPTR,GATTR.TYPTR) THEN
3021:                     GEN2(77(*CXP*),0(*SYS*),14(*FRDR*))
3022:                   ELSE
3023:                     IF COMPTYPES(CHARPTR,GATTR.TYPTR) THEN
3024:                       GEN2(77(*CXP*),0(*SYS*),16(*FRDC*))
3025:                     ELSE
3026:                       IF STRGTYPE(GATTR.TYPTR) THEN
3027:                         BEGIN GENLDC(GATTR.TYPTR^.MAXLENG);
3028:                           GEN2(77(*CXP*),0(*SYS*),18(*FRDS*))
3029:                         END
3030:                       ELSE ERROR(125);
3031:               IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
3032:               TEST := SY <> COMMA;
3033:               IF NOT TEST THEN INSYMBOL
3034:             UNTIL TEST
3035:           END;
3036:         IF LKEY = 2 THEN
```

```
3037:          BEGIN LOADIDADDR(FILEPTR);
3038:            GEN2(77(*CXP*),0(*SYS*),21(*FRLN*));
3039:            IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
3040:          END
3041:      END (*READ*) ;
3042:
3043:      PROCEDURE WRITE;
3044:        VAR LSP: STP; DEFAULT: BOOLEAN;
3045:            FILEPTR,LCP: CTP; LEN,LMIN,LMAX: INTEGER;
3046:      BEGIN FILEPTR := OUTPUTPTR;
3047:        IF (SY = IDENT) AND WASLPARENT THEN
3048:          BEGIN SEARCHID([FIELD,VARS,KONST,FUNC],LCP);
3049:            IF LCP^.IDTYPE <> NIL THEN
3050:              IF LCP^.IDTYPE^.FORM = FILES THEN
3051:                IF LCP^.IDTYPE^.FILTYPE = CHARPTR THEN
3052:                  BEGIN INSYMBOL; FILEPTR := LCP;
3053:                    IF NOT (SY IN [COMMA,RPARENT]) THEN ERROR(20);
3054:                    IF SY = COMMA THEN INSYMBOL
3055:                  END
3056:          END;
3057:        IF WASLPARENT AND (SY <> RPARENT) THEN
3058:          BEGIN
3059:            REPEAT LOADIDADDR(FILEPTR);
3060:              EXPRESSION(FSYS + [COMMA,COLON,RPARENT]);
3061:              LSP := GATTR.TYPTR;
3062:              IF LSP <> NIL THEN
3063:                IF LSP^.FORM <= SUBRANGE THEN LOAD
3064:                ELSE LOADADDRESS;
3065:              IF SY = COLON THEN
3066:                BEGIN INSYMBOL;
3067:                  EXPRESSION(FSYS + [COMMA,COLON,RPARENT]);
3068:                  IF GATTR.TYPTR <> NIL THEN
3069:                    IF GATTR.TYPTR <> INTPTR THEN ERROR(20);
3070:                  LOAD; DEFAULT := FALSE
3071:                END
3072:              ELSE DEFAULT := TRUE;
3073:              IF LSP = INTPTR THEN
3074:                BEGIN IF DEFAULT THEN GENLDC(0);
3075:                  GEN2(77(*CXP*),0(*SYS*),13(*FWRI*))
3076:                END
3077:              ELSE
3078:                IF LSP = REALPTR THEN
3079:                  BEGIN IF DEFAULT THEN GENLDC(0);
3080:                    IF SY = COLON THEN
3081:                      BEGIN INSYMBOL;
3082:                        EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
3083:                        IF GATTR.TYPTR <> NIL THEN
3084:                          IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
3085:                      END
3086:                    ELSE GENLDC(0);
3087:                    GEN2(77(*CXP*),0(*SYS*),15(*FWRR*))
3088:                  END
3089:                ELSE
3090:                  IF LSP = CHARPTR THEN
3091:                    BEGIN IF DEFAULT THEN GENLDC(0);
3092:                      GEN2(77(*CXP*),0(*SYS*),17(*FWRC*))
3093:                    END
3094:                  ELSE
3095:                    IF STRGTYPE(LSP) THEN
3096:                      BEGIN IF DEFAULT THEN GENLDC(0);
3097:                        GEN2(77(*CXP*),0(*SYS*),19(*FWRS*))
3098:                      END
3099:                    ELSE
3100:                      IF PAOFCHAR(LSP) THEN
3101:                        BEGIN LMAX := 0;
3102:                          IF LSP^.INXTYPE <> NIL THEN
```

```
3103:                           BEGIN GETBOUNDS(LSP^.INXTYPE,LMIN,LMAX);
3104:                              LMAX := LMAX - LMIN + 1
3105:                           END;
3106:                         IF DEFAULT THEN GENLDC(LMAX);
3107:                         GENLDC(LMAX);
3108:                         GEN2(77(*CXP*),0(*SYS*),20(*FWRB*))
3109:                       END
3110:                     ELSE ERROR(125);
3111:               IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
3112:               TEST := SY <> COMMA;
3113:               IF NOT TEST THEN INSYMBOL
3114:             UNTIL TEST;
3115:           END;
3116:         IF LKEY = 4 THEN (*WRITELN*)
3117:           BEGIN LOADIDADDR(FILEPTR);
3118:             GEN2(77(*CXP*),0(*SYS*),22(*FWLN*));
3119:             IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
3120:           END
3121:       END (*WRITE*) ;
3122:
3123:       PROCEDURE CALLNONSPECIAL;
3124:         VAR NXT,LCP: CTP; LSP: STP; LB: BOOLEAN;
3125:             LMIN,LMAX: INTEGER;
3126:       BEGIN
3127:         WITH FCP^ DO
3128:           BEGIN NXT := NEXT;
3129:             IF PFDECKIND = DECLARED THEN
3130:               IF PFKIND <> ACTUAL THEN ERROR(400)
3131:           END;
3132:         IF SY = LPARENT THEN
3133:           BEGIN
3134:             REPEAT
3135:               IF NXT = NIL THEN ERROR(126);
3136:               INSYMBOL;
3137:               EXPRESSION(FSYS + [COMMA,RPARENT]);
3138:               IF (GATTR.TYPTR <> NIL) AND (NXT <> NIL) THEN
3139:                 BEGIN LSP := NXT^.IDTYPE;
3140:                   IF LSP <> NIL THEN
3141:                     BEGIN
3142:                       IF NXT^.VKIND = ACTUAL THEN
3143:                         IF GATTR.TYPTR^.FORM <= POWER THEN
3144:                           BEGIN LB := (GATTR.TYPTR = CHARPTR)
3145:                                       AND (GATTR.KIND = CST);
3146:                             LOAD;
3147:                             IF LSP^.FORM = POWER THEN
3148:                               GEN1(32(*ADJ*),LSP^.SIZE)
3149:                             ELSE
3150:                             IF (LSP^.FORM = SUBRANGE)
3151:                                   AND RANGECHECK THEN
3152:                               BEGIN GENLDC(LSP^.MIN.IVAL);
3153:                                 GENLDC(LSP^.MAX.IVAL);
3154:                                 GEN0(8(*CHK*))
3155:                               END
3156:                             ELSE
3157:                             IF (GATTR.TYPTR = INTPTR) AND
3158:                                   COMPTYPES(LSP,REALPTR) THEN
3159:                               BEGIN GEN0(10(*FLT*));
3160:                                 GATTR.TYPTR := REALPTR
3161:                               END
3162:                             ELSE
3163:                             IF LB AND STRGTYPE(LSP) THEN
3164:                               GATTR.TYPTR := STRGPTR
3165:                           END
3166:                         ELSE (*FORM > POWER*)
3167:                           BEGIN LB := STRGTYPE(GATTR.TYPTR)
3168:                                       AND (GATTR.KIND = CST);
```

```
3169:                           LOADADDRESS;
3170:                           IF LB AND PAOFCHAR(LSP) THEN
3171:                             IF NOT LSP^.AISSTRNG THEN
3172:                               BEGIN GEN0(80(*S1P*));
3173:                                 IF LSP^.INXTYPE <> NIL THEN
3174:                                   BEGIN
3175:                                     GETBOUNDS(LSP^.INXTYPE,LMIN,LMAX);
3176:                                     IF LMAX-LMIN+1 <>
3177:                                         GATTR.TYPTR^.MAXLENG THEN ERROR(142);
3178:                                   END;
3179:                                 GATTR.TYPTR := LSP
3180:                               END
3181:                         END
3182:                       ELSE (*VKIND = FORMAL*)
3183:                         IF GATTR.KIND = VARBL THEN
3184:                           BEGIN
3185:                             IF GATTR.ACCESS = BYTE THEN ERROR(103);
3186:                             LOADADDRESS;
3187:                             IF (LSP^.FORM=POWER) THEN
3188:                               IF GATTR.TYPTR^.SIZE <>
3189:                                   LSP^.SIZE THEN ERROR(142)
3190:                           END
3191:                         ELSE ERROR(154);
3192:                       IF NOT COMPTYPES(LSP,GATTR.TYPTR) THEN ERROR(142)
3193:                     END
3194:                 END;
3195:               IF NXT <> NIL THEN NXT := NXT^.NEXT
3196:             UNTIL SY <> COMMA;
3197:             IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3198:           END (*LPARENT*) ;
3199:         IF NXT <> NIL THEN ERROR(126);
3200:         WITH FCP^ DO
3201:           IF PFDECKIND = DECLARED THEN
3202:             BEGIN
3203:               IF KLASS = FUNC THEN
3204:                 BEGIN GENLDC(0); GENLDC(0) END;
3205:               IF PFSEG <> SEG THEN GEN2(77(*CXP*),PFSEG,PFNAME)
3206:               ELSE
3207:                 IF PFLEV = 0 THEN GEN1(66(*CBP*),PFNAME)
3208:                 ELSE
3209:                   IF PFLEV = LEVEL THEN GEN1(78(*CLP*),PFNAME)
3210:                   ELSE
3211:                   IF PFLEV = 1 THEN GEN1(79(*CGP*),PFNAME)
3212:                   ELSE GEN1(46(*CIP*),PFNAME)
3213:             END
3214:           ELSE
3215:             IF (CSPNUM <> 21) AND (CSPNUM <> 22) THEN
3216:               GEN1(30(*CSP*),CSPNUM);
3217:         GATTR.TYPTR := FCP^.IDTYPE
3218:       END (*CALLNONSPECIAL*) ;
3219:
3220:     BEGIN (*CALL*)
3221:       IF FCP^.PFDECKIND = SPECIAL THEN
3222:         BEGIN WASLPARENT := TRUE; LKEY := FCP^.KEY;
3223:           IF SY = LPARENT THEN INSYMBOL
3224:           ELSE
3225:             IF LKEY IN [2,4,5,6] THEN WASLPARENT := FALSE
3226:             ELSE ERROR(9);
3227:           IF LKEY IN [7,8,9,10,11,13,14,25,36] THEN
3228:             BEGIN EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD END;
3229:           CASE LKEY OF
3230:              1,2: READ;
3231:              3,4: WRITE;
3232:              5,6: BEGIN (*EOF & EOLN*)
3233:                     IF WASLPARENT THEN
3234:                       BEGIN VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
```

```
3235:                       IF GATTR.TYPTR <> NIL THEN
3236:                         IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
3237:                         ELSE
3238:                           IF (GATTR.TYPTR^.FILTYPE <> CHARPTR) AND
3239:                               (LKEY = 6) THEN ERROR(399)
3240:                     END
3241:                   ELSE
3242:                     LOADIDADDR(INPUTPTR);
3243:                   GENLDC(0); GENLDC(0);
3244:                   IF LKEY = 5 THEN GEN2(77(*CXP*),0(*SYS*),10(*FEOF*))
3245:                   ELSE GEN2(77(*CXP*),0(*SYS*),11(*FEOLN*));
3246:                   GATTR.TYPTR := BOOLPTR
3247:                 END (*EOF*) ;
3248:            7,8: BEGIN GENLDC(1); (*PREDSUCC*)
3249:                   IF GATTR.TYPTR <> NIL THEN
3250:                     IF GATTR.TYPTR^.FORM = SCALAR THEN
3251:                       IF LKEY = 8 THEN GEN0(2(*ADI*))
3252:                       ELSE GEN0(21(*SBI*))
3253:                     ELSE ERROR(115)
3254:                 END (*PREDSUCC*) ;
3255:              9: BEGIN (*ORD*)
3256:                   IF GATTR.TYPTR <> NIL THEN
3257:                     IF GATTR.TYPTR^.FORM >= POWER THEN ERROR(125);
3258:                   GATTR.TYPTR := INTPTR
3259:                 END (*ORD*) ;
3260:             10: BEGIN (*SQR*)
3261:                   IF GATTR.TYPTR <> NIL THEN
3262:                     IF GATTR.TYPTR = INTPTR THEN GEN0(24(*SQI*))
3263:                     ELSE
3264:                       IF GATTR.TYPTR = REALPTR THEN GEN0(25(*SQR*))
3265:                       ELSE BEGIN ERROR(125); GATTR.TYPTR := INTPTR END
3266:                 END (*SQR*) ;
3267:             11: BEGIN (*ABS*)
3268:                   IF GATTR.TYPTR <> NIL THEN
3269:                     IF GATTR.TYPTR = INTPTR THEN GEN0(0(*ABI*))
3270:                     ELSE
3271:                       IF GATTR.TYPTR = REALPTR THEN GEN0(1(*ABR*))
3272:                       ELSE BEGIN ERROR(125); GATTR.TYPTR := INTPTR END
3273:                 END (*ABS*) ;
3274:             12: NEWSTMT;
3275:          13,14: UNITIO;
3276:             15: CONCAT;
3277:             16: BEGIN (*LENGTH*)
3278:                   STRGVAR(FSYS + [RPARENT],FALSE);
3279:                   GEN0(62(*LDB*)); GATTR.TYPTR := INTPTR
3280:                 END (*LENGTH*) ;
3281:             17: BEGIN (*INSERT*)
3282:                   STRGVAR(FSYS + [COMMA],FALSE);
3283:                   IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3284:                   STRGVAR(FSYS + [COMMA],TRUE);
3285:                   GENLDC(GATTR.TYPTR^.MAXLENG);
3286:                   IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3287:                   EXPRESSION(FSYS + [RPARENT]); LOAD;
3288:                   IF GATTR.TYPTR <> NIL THEN
3289:                     IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
3290:                   GEN2(77(*CXP*),0(*SYS*),24(*SINSERT*))
3291:                 END (*INSERT*) ;
3292:       43,18,19: COPYDELETE;
3293:             20: BEGIN (*POS*)
3294:                   STRGVAR(FSYS + [COMMA],FALSE);
3295:                   IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3296:                   STRGVAR(FSYS + [RPARENT],FALSE);
3297:                   GENLDC(0); GENLDC(0);
3298:                   GEN2(77(*CXP*),0(*SYS*),27(*SPOS*));
3299:                   GATTR.TYPTR := INTPTR
3300:                 END (*POS*) ;
```

```
3301:        27,21,22: MOVE;
3302:              23: EXIT;
3303:              24: BEGIN (*IDSEARCH*)
3304:                    VARIABLE(FSYS + [COMMA]); LOADADDRESS;
3305:                    IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3306:                    VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3307:                    GEN1(30(*CSP*),7(*IDS*))
3308:                  END (*IDSEARCH*) ;
3309:              25: BEGIN (*TREESEARCH*)
3310:                    IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3311:                    VARIABLE(FSYS + [COMMA]); LOADADDRESS;
3312:                    IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3313:                    VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3314:                    GATTR.TYPTR := INTPTR;
3315:                    GEN1(30(*CSP*),8(*TRS*))
3316:                  END (*TREESEARCH*) ;
3317:              26: BEGIN (*TIME*)
3318:                    VARIABLE(FSYS + [COMMA]); LOADADDRESS;
3319:                    IF GATTR.TYPTR <> NIL THEN
3320:                      IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
3321:                    IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3322:                    VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3323:                    IF GATTR.TYPTR <> NIL THEN
3324:                      IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
3325:                    GEN1(30(*CSP*),9(*TIM*))
3326:                  END (*TIME*) ;
3327:     33,28,29,30: BEGIN (*OPEN,RESET,REWRITE*)
3328:                    VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
3329:                    IF GATTR.TYPTR <> NIL THEN
3330:                      IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125);
3331:                    IF SY <> COMMA THEN
3332:                      IF LKEY = 33 THEN
3333:                        GEN2(77(*CXP*),0(*SYS*),4(*FRESET*))
3334:                      ELSE ERROR(20)
3335:                    ELSE
3336:                      BEGIN INSYMBOL;
3337:                        STRGVAR(FSYS + [RPARENT],FALSE);
3338:                        IF (LKEY = 28) OR (LKEY = 30) THEN
3339:                          GENLDC(0)
3340:                        ELSE GENLDC(1);
3341:                        GENLDC(0); GEN2(77(*CXP*),0(*SYS*),5(*FOPEN*))
3342:                      END;
3343:                    IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
3344:                  END (*OPEN*) ;
3345:              31: CLOSE;
3346:     32,34,35,40: GETPUTETC;
3347:              36: SCAN;
3348:           37,38: BLOCKIO;
3349:           39,42: DRAWSTUFF;
3350:              41: SIZEOF
3351:           END (*SPECIAL CASES*) ;
3352:           IF WASLPARENT THEN
3353:             IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3354:         END (*SPECIAL PROCEDURES AND FUNCTIONS*)
3355:       ELSE CALLNONSPECIAL
3356:     END (*CALL*) ;
3357:    (*    COPYRIGHT (C) 1978, REGENTS OF THE       *)
3358:   (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
3359:
3360:     PROCEDURE EXPRESSION(*FSYS: SETOFSYS*);
3361:       LABEL 1;    (* STRING COMPARE KLUDGE *)
3362:       VAR LATTR: ATTR; LOP: OPERATOR; TYPIND: INTEGER;
3363:           LSIZE: ADDRRANGE; LSTRING,GSTRING: BOOLEAN;
3364:           LMIN,LMAX: INTEGER;
3365:
3366:       PROCEDURE FLOATIT(VAR FSP: STP);
```

```
3367:      BEGIN
3368:        IF GATTR.TYPTR = INTPTR THEN
3369:          BEGIN GEN0(10(*FLT*)); GATTR.TYPTR := REALPTR END;
3370:        IF FSP = INTPTR THEN
3371:          BEGIN GEN0(9(*FLO*)); FSP := REALPTR END
3372:      END (*FLOATIT*) ;
3373:
3374:      PROCEDURE SIMPLEEXPRESSION(FSYS: SETOFSYS);
3375:        VAR LATTR: ATTR; LOP: OPERATOR; SIGNED: BOOLEAN;
3376:
3377:        PROCEDURE TERM(FSYS: SETOFSYS);
3378:          VAR LATTR: ATTR; LOP: OPERATOR;
3379:
3380:          PROCEDURE FACTOR(FSYS: SETOFSYS);
3381:            VAR LCP: CTP; LVP: CSP; VARPART,ALLCONST: BOOLEAN;
3382:                LSP: STP; HIGHVAL,LOWVAL,LIC,LOP: INTEGER;
3383:                CSTPART: SET OF 0..127;
3384:          BEGIN
3385:            IF NOT (SY IN FACBEGSYS) THEN
3386:              BEGIN ERROR(58); SKIP(FSYS + FACBEGSYS);
3387:                GATTR.TYPTR := NIL
3388:              END;
3389:            WHILE SY IN FACBEGSYS DO
3390:              BEGIN
3391:                CASE SY OF
3392:          (*ID*)  IDENT:
3393:                    BEGIN SEARCHID([KONST,VARS,FIELD,FUNC],LCP); INSYMBOL;
3394:                      IF LCP^.KLASS = FUNC THEN
3395:                        BEGIN CALL(FSYS,LCP); GATTR.KIND := EXPR END
3396:                      ELSE
3397:                        IF LCP^.KLASS = KONST THEN
3398:                          WITH GATTR, LCP^ DO
3399:                            BEGIN TYPTR := IDTYPE; KIND := CST;
3400:                              CVAL := VALUES
3401:                            END
3402:                        ELSE SELECTOR(FSYS,LCP);
3403:                      IF GATTR.TYPTR <> NIL THEN
3404:                        WITH GATTR,TYPTR^ DO
3405:                          IF FORM = SUBRANGE THEN TYPTR := RANGETYPE
3406:                    END;
3407:          (*CST*) INTCONST:
3408:                    BEGIN
3409:                      WITH GATTR DO
3410:                        BEGIN TYPTR := INTPTR; KIND := CST;
3411:                          CVAL := VAL
3412:                        END;
3413:                      INSYMBOL
3414:                    END;
3415:                  REALCONST:
3416:                    BEGIN
3417:                      WITH GATTR DO
3418:                        BEGIN TYPTR := REALPTR; KIND := CST;
3419:                          CVAL := VAL
3420:                        END;
3421:                      INSYMBOL
3422:                    END;
3423:                  STRINGCONST:
3424:                    BEGIN
3425:                      WITH GATTR DO
3426:                        BEGIN
3427:                          IF LGTH = 1 THEN TYPTR := CHARPTR
3428:                          ELSE
3429:                            BEGIN NEW(LSP,ARRAYS,TRUE,TRUE);
3430:                              LSP^ := STRGPTR^;
3431:                              LSP^.MAXLENG := LGTH;
3432:                              TYPTR := LSP
```

```
3433:                         END;
3434:                       KIND := CST; CVAL := VAL
3435:                     END;
3436:                   INSYMBOL
3437:                 END;
3438:         (*(*)   LPARENT:
3439:                 BEGIN INSYMBOL; EXPRESSION(FSYS + [RPARENT]);
3440:                   IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3441:                 END;
3442:         (*NOT*) NOTSY:
3443:                 BEGIN INSYMBOL; FACTOR(FSYS);
3444:                   LOAD; GEN0(19(*NOT*));
3445:                   IF GATTR.TYPTR <> NIL THEN
3446:                     IF GATTR.TYPTR <> BOOLPTR THEN
3447:                       BEGIN ERROR(135); GATTR.TYPTR := NIL END;
3448:                 END;
3449:         (*[*)   LBRACK:
3450:                 BEGIN INSYMBOL; CSTPART := [ ]; VARPART := FALSE;
3451:                   NEW(LSP,POWER);
3452:                   WITH LSP^ DO
3453:                     BEGIN ELSET := NIL; SIZE := 0; FORM := POWER END;
3454:                   IF SY = RBRACK THEN
3455:                     BEGIN
3456:                       WITH GATTR DO
3457:                         BEGIN TYPTR := LSP; KIND := CST END;
3458:                       INSYMBOL
3459:                     END
3460:                   ELSE
3461:                     BEGIN
3462:                       REPEAT EXPRESSION(FSYS + [COMMA,RBRACK,COLON]);
3463:                         IF GATTR.TYPTR <> NIL THEN
3464:                           IF GATTR.TYPTR^.FORM <> SCALAR THEN
3465:                             BEGIN ERROR(136); GATTR.TYPTR := NIL END
3466:                           ELSE
3467:                             IF COMPTYPES(LSP^.ELSET,GATTR.TYPTR) THEN
3468:                               BEGIN ALLCONST := FALSE; LOP := 23(*SGS*);
3469:                                 IF (GATTR.KIND = CST) AND
3470:                                    (GATTR.CVAL.IVAL <= 127) THEN
3471:                                   BEGIN ALLCONST := TRUE;
3472:                                     LOWVAL := GATTR.CVAL.IVAL;
3473:                                     HIGHVAL := LOWVAL
3474:                                   END;
3475:                                 LIC := IC; LOAD;
3476:                                 IF SY = COLON THEN
3477:                                   BEGIN INSYMBOL; LOP := 20(*SRS*);
3478:                                     EXPRESSION(FSYS + [COMMA,RBRACK]);
3479:                                     IF COMPTYPES(LSP^.ELSET,GATTR.TYPTR) THEN
3480:                                     ELSE
3481:                                       BEGIN ERROR(137); GATTR.TYPTR:=NIL END;
3482:                                     IF ALLCONST THEN
3483:                                       IF (GATTR.KIND = CST) AND
3484:                                          (GATTR.CVAL.IVAL <= 127) THEN
3485:                                           HIGHVAL := GATTR.CVAL.IVAL
3486:                                       ELSE
3487:                                         BEGIN LOAD; ALLCONST := FALSE END
3488:                                     ELSE LOAD
3489:                                   END;
3490:                                 IF ALLCONST THEN
3491:                                   BEGIN IC := LIC; (*FORGET FIRST CONST*)
3492:                                     CSTPART := CSTPART + [LOWVAL..HIGHVAL]
3493:                                   END
3494:                                 ELSE
3495:                                   BEGIN GEN0(LOP);
3496:                                     IF VARPART THEN GEN0(28(*UNI*))
3497:                                     ELSE VARPART := TRUE
3498:                                   END;
```

```
3499:                                 LSP^.ELSET := GATTR.TYPTR;
3500:                                 GATTR.TYPTR := LSP
3501:                               END
3502:                             ELSE ERROR(137);
3503:                           TEST := SY <> COMMA;
3504:                           IF NOT TEST THEN INSYMBOL
3505:                         UNTIL TEST;
3506:                         IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
3507:                       END;
3508:                     IF VARPART THEN
3509:                       BEGIN
3510:                         IF CSTPART <> [ ] THEN
3511:                           BEGIN
3512:                             SCONST^.PVAL := CSTPART;
3513:                             SCONST^.CCLASS := PSET;
3514:                             GATTR.CVAL.VALP := SCONST;
3515:                             GATTR.KIND := CST;
3516:                             LOAD; GEN0(28(*UNI*))
3517:                           END;
3518:                         GATTR.KIND := EXPR
3519:                       END
3520:                     ELSE
3521:                       BEGIN
3522:                         SCONST^.PVAL := CSTPART;
3523:                         SCONST^.CCLASS := PSET;
3524:                         GATTR.CVAL.VALP := SCONST;
3525:                         GATTR.KIND := CST
3526:                       END
3527:                   END
3528:               END (*CASE*) ;
3529:               IF NOT (SY IN FSYS) THEN
3530:                 BEGIN ERROR(6); SKIP(FSYS + FACBEGSYS) END
3531:             END (*WHILE*)
3532:         END (*FACTOR*) ;
3533:
3534:       BEGIN (*TERM*)
3535:         FACTOR(FSYS + [MULOP]);
3536:         WHILE SY = MULOP DO
3537:           BEGIN LOAD; LATTR := GATTR; LOP := OP;
3538:             INSYMBOL; FACTOR(FSYS + [MULOP]); LOAD;
3539:             IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3540:               CASE LOP OF
3541:       (***)     MUL:  IF (LATTR.TYPTR = INTPTR) AND (GATTR.TYPTR = INTPTR)
3542:                       THEN GEN0(15(*MPI*))
3543:                     ELSE
3544:                       BEGIN FLOATIT(LATTR.TYPTR);
3545:                         IF (LATTR.TYPTR = REALPTR) AND
3546:                            (GATTR.TYPTR = REALPTR) THEN GEN0(16(*MPR*))
3547:                         ELSE
3548:                           IF (LATTR.TYPTR^.FORM = POWER)
3549:                               AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3550:                             GEN0(12(*INT*))
3551:                           ELSE BEGIN ERROR(134); GATTR.TYPTR:=NIL END
3552:                       END;
3553:       (*/*)     RDIV: BEGIN FLOATIT(LATTR.TYPTR);
3554:                       IF (LATTR.TYPTR = REALPTR) AND
3555:                          (GATTR.TYPTR = REALPTR) THEN GEN0(7(*DVR*))
3556:                       ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3557:                     END;
3558:       (*DIV*)   IDIV: IF (LATTR.TYPTR = INTPTR) AND
3559:                          (GATTR.TYPTR = INTPTR) THEN GEN0(6(*DVI*))
3560:                     ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END;
3561:       (*MOD*)   IMOD: IF (LATTR.TYPTR = INTPTR) AND
3562:                          (GATTR.TYPTR = INTPTR) THEN GEN0(14(*MOD*))
3563:                     ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END;
3564:       (*AND*)   ANDOP:IF (LATTR.TYPTR = BOOLPTR) AND
```

```
3565:                          (GATTR.TYPTR = BOOLPTR) THEN GEN0(4(*AND*))
3566:                       ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3567:               END (*CASE*)
3568:             ELSE GATTR.TYPTR := NIL
3569:           END (*WHILE*)
3570:       END (*TERM*) ;
3571:
3572:     BEGIN (*SIMPLEEXPRESSION*)
3573:       SIGNED := FALSE;
3574:       IF (SY = ADDOP) AND (OP IN [PLUS,MINUS]) THEN
3575:         BEGIN SIGNED := OP = MINUS; INSYMBOL END;
3576:       TERM(FSYS + [ADDOP]);
3577:       IF SIGNED THEN
3578:         BEGIN LOAD;
3579:           IF GATTR.TYPTR = INTPTR THEN GEN0(17(*NGI*))
3580:           ELSE
3581:             IF GATTR.TYPTR = REALPTR THEN GEN0(18(*NGR*))
3582:             ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3583:         END;
3584:       WHILE SY = ADDOP DO
3585:         BEGIN LOAD; LATTR := GATTR; LOP := OP;
3586:           INSYMBOL; TERM(FSYS + [ADDOP]); LOAD;
3587:           IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3588:             CASE LOP OF
3589:     (*+*)     PLUS:
3590:                 IF (LATTR.TYPTR = INTPTR)AND(GATTR.TYPTR = INTPTR) THEN
3591:                   GEN0(2(*ADI*))
3592:                 ELSE
3593:                   BEGIN FLOATIT(LATTR.TYPTR);
3594:                     IF (LATTR.TYPTR = REALPTR)AND(GATTR.TYPTR = REALPTR)
3595:                       THEN GEN0(3(*ADR*))
3596:                     ELSE IF (LATTR.TYPTR^.FORM = POWER)
3597:                              AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3598:                            GEN0(28(*UNI*))
3599:                          ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3600:                   END;
3601:     (*-*)     MINUS:
3602:                 IF (LATTR.TYPTR = INTPTR) AND (GATTR.TYPTR = INTPTR) THEN
3603:                   GEN0(21(*SBI*))
3604:                 ELSE
3605:                   BEGIN FLOATIT(LATTR.TYPTR);
3606:                     IF (LATTR.TYPTR = REALPTR) AND (GATTR.TYPTR = REALPTR)
3607:                       THEN GEN0(22(*SBR*))
3608:                     ELSE
3609:                       IF (LATTR.TYPTR^.FORM = POWER)
3610:                           AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3611:                         GEN0(5(*DIF*))
3612:                       ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3613:                   END;
3614:     (*OR*)    OROP:
3615:                 IF (LATTR.TYPTR = BOOLPTR) AND (GATTR.TYPTR = BOOLPTR) THEN
3616:                   GEN0(13(*IOR*))
3617:                 ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3618:             END (*CASE*)
3619:           ELSE GATTR.TYPTR := NIL
3620:         END (*WHILE*)
3621:     END (*SIMPLEEXPRESSION*) ;
3622:
3623:     PROCEDURE MAKEPA(VAR STRGFSP: STP; PAFSP: STP);
3624:       VAR LMIN,LMAX: INTEGER;
3625:     BEGIN
3626:       IF PAFSP^.INXTYPE <> NIL THEN
3627:         BEGIN GETBOUNDS(PAFSP^.INXTYPE,LMIN,LMAX);
3628:           IF LMAX-LMIN+1 <> STRGFSP^.MAXLENG THEN ERROR(129)
3629:         END;
3630:       STRGFSP := PAFSP
```

```
3631:      END (*MAKEPA*) ;
3632:
3633:    BEGIN (*EXPRESSION*)
3634:      SIMPLEEXPRESSION(FSYS + [RELOP]);
3635:      IF SY = RELOP THEN
3636:        BEGIN
3637:          LSTRING := (GATTR.KIND = CST) AND
3638:                  (STRGTYPE(GATTR.TYPTR) OR (GATTR.TYPTR = CHARPTR));
3639:          IF GATTR.TYPTR <> NIL THEN
3640:            IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3641:            ELSE LOADADDRESS;
3642:          LATTR := GATTR; LOP := OP;
3643:          INSYMBOL; SIMPLEEXPRESSION(FSYS);
3644:          GSTRING := (GATTR.KIND = CST) AND
3645:                  (STRGTYPE(GATTR.TYPTR) OR (GATTR.TYPTR = CHARPTR));
3646:          IF GATTR.TYPTR <> NIL THEN
3647:            IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3648:            ELSE LOADADDRESS;
3649:          IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3650:            IF LOP = INOP THEN
3651:              IF GATTR.TYPTR^.FORM = POWER THEN
3652:                IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR^.ELSET) THEN
3653:                  GEN0(11(*INN*))
3654:                ELSE BEGIN ERROR(129); GATTR.TYPTR := NIL END
3655:              ELSE BEGIN ERROR(130); GATTR.TYPTR := NIL END
3656:            ELSE
3657:              BEGIN
3658:                IF LATTR.TYPTR <> GATTR.TYPTR THEN
3659:                  FLOATIT(LATTR.TYPTR);
3660:                IF LSTRING THEN
3661:                  BEGIN
3662:                    IF PAOFCHAR(GATTR.TYPTR) THEN
3663:                      IF NOT GATTR.TYPTR^.AISSTRNG THEN
3664:                        BEGIN GEN0(29(*S2P*));
3665:                          MAKEPA(LATTR.TYPTR,GATTR.TYPTR)
3666:                        END
3667:                  END
3668:                ELSE
3669:                  IF GSTRING THEN
3670:                    BEGIN
3671:                      IF PAOFCHAR(LATTR.TYPTR) THEN
3672:                        IF NOT LATTR.TYPTR^.AISSTRNG THEN
3673:                          BEGIN GEN0(80(*S1P*));
3674:                            MAKEPA(GATTR.TYPTR,LATTR.TYPTR)
3675:                          END;
3676:                    END;
3677:                IF (LSTRING AND STRGTYPE(GATTR.TYPTR)) OR
3678:                   (GSTRING AND STRGTYPE(LATTR.TYPTR)) THEN GOTO 1;
3679:                IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3680:                  BEGIN LSIZE := LATTR.TYPTR^.SIZE;
3681:                    CASE LATTR.TYPTR^.FORM OF
3682:                      SCALAR:
3683:                        IF LATTR.TYPTR = REALPTR THEN TYPIND := 1
3684:                        ELSE
3685:                          IF LATTR.TYPTR = BOOLPTR THEN TYPIND := 3
3686:                          ELSE TYPIND := 0;
3687:                      POINTER:
3688:                        BEGIN
3689:                          IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131);
3690:                          TYPIND := 0
3691:                        END;
3692:                      POWER:
3693:                        BEGIN
3694:                          IF LOP IN [LTOP,GTOP] THEN ERROR(132);
3695:                          TYPIND := 4
3696:                        END;
```

```
3697:                       ARRAYS:
3698:                         BEGIN
3699:                           TYPIND := 6;
3700:                           IF PAOFCHAR(LATTR.TYPTR) THEN
3701:                             IF LATTR.TYPTR^.AISSTRNG THEN
3702:                 1:            TYPIND := 2
3703:                             ELSE
3704:                               BEGIN TYPIND := 5;
3705:                                 IF LATTR.TYPTR^.INXTYPE <> NIL THEN
3706:                                   BEGIN
3707:                                     GETBOUNDS(LATTR.TYPTR^.INXTYPE,LMIN,LMAX);
3708:                                     LSIZE := LMAX - LMIN + 1
3709:                                   END
3710:                               END
3711:                           ELSE
3712:                             IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131)
3713:                         END;
3714:                       RECORDS:
3715:                         BEGIN
3716:                           IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131);
3717:                           TYPIND := 6
3718:                         END;
3719:                       FILES:
3720:                         BEGIN ERROR(133); TYPIND := 0 END
3721:                     END;
3722:                     CASE LOP OF
3723:                       LTOP: GEN2(53(*LES*),TYPIND,LSIZE);
3724:                       LEOP: GEN2(52(*LEQ*),TYPIND,LSIZE);
3725:                       GTOP: GEN2(49(*GRT*),TYPIND,LSIZE);
3726:                       GEOP: GEN2(48(*GEQ*),TYPIND,LSIZE);
3727:                       NEOP: GEN2(55(*NEQ*),TYPIND,LSIZE);
3728:                       EQOP: GEN2(47(*EQU*),TYPIND,LSIZE)
3729:                     END
3730:                   END
3731:                 ELSE ERROR(129)
3732:               END;
3733:           GATTR.TYPTR := BOOLPTR; GATTR.KIND := EXPR
3734:         END (*SY = RELOP*)
3735:     END (*EXPRESSION*) ;
3736:
3737:     PROCEDURE STATEMENT(FSYS: SETOFSYS);
3738:       LABEL 1;
3739:       VAR LCP: CTP; TTOP: DISPRANGE; LLP: LABELP; HEAP: ^INTEGER;
3740:
3741:       PROCEDURE ASSIGNMENT(FCP: CTP);
3742:         VAR LATTR: ATTR; CSTRING,PAONLEFT: BOOLEAN; LMIN,LMAX: INTEGER;
3743:       BEGIN SELECTOR(FSYS + [BECOMES],FCP);
3744:         IF SY = BECOMES THEN
3745:           BEGIN LMAX := 0; CSTRING := FALSE;
3746:             IF GATTR.TYPTR <> NIL THEN
3747:               IF (GATTR.ACCESS = INDRCT) OR (GATTR.TYPTR^.FORM > POWER) THEN
3748:                 LOADADDRESS;
3749:             PAONLEFT := PAOFCHAR(GATTR.TYPTR);
3750:             LATTR := GATTR;
3751:             INSYMBOL; EXPRESSION(FSYS);
3752:             IF GATTR.KIND = CST THEN
3753:               CSTRING := (GATTR.TYPTR = CHARPTR) OR STRGTYPE(GATTR.TYPTR);
3754:             IF GATTR.TYPTR <> NIL THEN
3755:               IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3756:               ELSE LOADADDRESS;
3757:             IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3758:               BEGIN
3759:                 IF GATTR.TYPTR = INTPTR THEN
3760:                   IF COMPTYPES(REALPTR,LATTR.TYPTR) THEN
3761:                     BEGIN GEN0(10(*FLT*)); GATTR.TYPTR := REALPTR END;
3762:               IF PAONLEFT THEN
```

```
3763:                 IF LATTR.TYPTR^.AISSTRNG THEN
3764:                   IF CSTRING AND (GATTR.TYPTR = CHARPTR) THEN
3765:                     GATTR.TYPTR := STRGPTR
3766:                   ELSE
3767:                 ELSE
3768:                   IF LATTR.TYPTR^.INXTYPE <> NIL THEN
3769:                     BEGIN GETBOUNDS(LATTR.TYPTR^.INXTYPE,LMIN,LMAX);
3770:                       LMAX := LMAX - LMIN + 1;
3771:                       IF CSTRING AND (GATTR.TYPTR <> CHARPTR) THEN
3772:                         BEGIN GEN0(80(*S1P*));
3773:                           IF LMAX <> GATTR.TYPTR^.MAXLENG THEN ERROR(129);
3774:                           GATTR.TYPTR := LATTR.TYPTR
3775:                         END
3776:                     END
3777:                   ELSE GATTR.TYPTR := LATTR.TYPTR;
3778:               IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3779:                 CASE LATTR.TYPTR^.FORM OF
3780:                   SUBRANGE: BEGIN
3781:                               IF RANGECHECK THEN
3782:                                 BEGIN
3783:                                   GENLDC(LATTR.TYPTR^.MIN.IVAL);
3784:                                   GENLDC(LATTR.TYPTR^.MAX.IVAL);
3785:                                   GEN0(8(*CHK*))
3786:                                 END;
3787:                               STORE(LATTR)
3788:                             END;
3789:                   POWER:    BEGIN
3790:                               GEN1(32(*ADJ*),LATTR.TYPTR^.SIZE);
3791:                               STORE(LATTR)
3792:                             END;
3793:                   SCALAR,
3794:                   POINTER: STORE(LATTR);
3795:                   ARRAYS:  IF PAONLEFT THEN
3796:                              IF LATTR.TYPTR^.AISSTRNG THEN
3797:                                GEN1(42(*SAS*),LATTR.TYPTR^.MAXLENG)
3798:                              ELSE GEN1(41(*MVB*),LMAX)
3799:                            ELSE GEN1(40(*MOV*),LATTR.TYPTR^.SIZE);
3800:                   RECORDS: GEN1(40(*MOV*),LATTR.TYPTR^.SIZE);
3801:                   FILES:   ERROR(146)
3802:                 END
3803:               ELSE ERROR(129)
3804:             END
3805:         END (*SY = BECOMES*)
3806:       ELSE ERROR(51)
3807:     END (*ASSIGNMENT*) ;
3808:
3809:     PROCEDURE GOTOSTATEMENT;
3810:       VAR LLP: LABELP; FOUND: BOOLEAN; TTOP: DISPRANGE;
3811:     BEGIN
3812:       IF NOT GOTOOK THEN ERROR(6);
3813:       IF SY = INTCONST THEN
3814:         BEGIN
3815:           FOUND := FALSE; TTOP := TOP;
3816:           WHILE DISPLAY[TTOP].OCCUR <> BLCK DO TTOP := TTOP - 1;
3817:           LLP := DISPLAY[TTOP].FLABEL;
3818:           WHILE (LLP <> NIL) AND NOT FOUND DO
3819:             WITH LLP^ DO
3820:               IF LABVAL = VAL.IVAL THEN
3821:                 BEGIN FOUND := TRUE;
3822:                   GENJMP(57(*UJP*),CODELBP)
3823:                 END
3824:               ELSE LLP := NEXTLAB;
3825:           IF NOT FOUND THEN ERROR(167);
3826:           INSYMBOL
3827:         END
3828:       ELSE ERROR(15)
```

```
3829:      END (*GOTOSTATEMENT*) ;
3830:
3831:      PROCEDURE COMPOUNDSTATEMENT;
3832:      BEGIN
3833:        REPEAT
3834:          REPEAT STATEMENT(FSYS + [SEMICOLON,ENDSY])
3835:          UNTIL NOT (SY IN STATBEGSYS);
3836:          TEST := SY <> SEMICOLON;
3837:          IF NOT TEST THEN INSYMBOL
3838:        UNTIL TEST;
3839:        IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
3840:      END (*COMPOUNDSTATEMENET*) ;
3841:
3842:      PROCEDURE IFSTATEMENT;
3843:        VAR LCIX1,LCIX2: LBP;
3844:      BEGIN EXPRESSION(FSYS + [THENSY]);
3845:        GENLABEL(LCIX1); GENFJP(LCIX1);
3846:        IF SY = THENSY THEN INSYMBOL ELSE ERROR(52);
3847:        STATEMENT(FSYS + [ELSESY]);
3848:        IF SY = ELSESY THEN
3849:          BEGIN GENLABEL(LCIX2); GENJMP(57(*UJP*),LCIX2);
3850:            PUTLABEL(LCIX1);
3851:            INSYMBOL; STATEMENT(FSYS);
3852:            PUTLABEL(LCIX2)
3853:          END
3854:        ELSE PUTLABEL(LCIX1)
3855:      END (*IFSTATEMENT*) ;
3856:
3857:      PROCEDURE CASESTATEMENT;
3858:        LABEL 1;
3859:        TYPE CIP = ^CASEINFO;
3860:             CASEINFO = RECORD
3861:                          NEXT: CIP;
3862:                          CSSTART: INTEGER;
3863:                          CSLAB: INTEGER
3864:                        END;
3865:        VAR LSP,LSP1: STP; FSTPTR,LPT1,LPT2,LPT3: CIP; LVAL: VALU;
3866:            LADDR, LCIX: LBP; NULSTMT, LMIN, LMAX: INTEGER;
3867:      BEGIN EXPRESSION(FSYS + [OFSY,COMMA,COLON]);
3868:        LOAD; GENLABEL(LCIX); GENJMP(57(*UJP*),LCIX);
3869:        LSP := GATTR.TYPTR;
3870:        IF LSP <> NIL THEN
3871:          IF (LSP^.FORM <> SCALAR) OR (LSP = REALPTR) THEN
3872:            BEGIN ERROR(144); LSP := NIL END;
3873:        IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
3874:        FSTPTR := NIL; GENLABEL(LADDR);
3875:        REPEAT
3876:          LPT3 := NIL;
3877:          REPEAT CONSTANT(FSYS + [COMMA,COLON],LSP1,LVAL);
3878:            IF LSP <> NIL THEN
3879:              IF COMPTYPES(LSP,LSP1) THEN
3880:                BEGIN LPT1 := FSTPTR; LPT2 := NIL;
3881:                  WHILE LPT1 <> NIL DO
3882:                    WITH LPT1^ DO
3883:                      BEGIN
3884:                        IF CSLAB <= LVAL.IVAL THEN
3885:                          BEGIN IF CSLAB = LVAL.IVAL THEN ERROR(156);
3886:                            GOTO 1
3887:                          END;
3888:                        LPT2 := LPT1; LPT1 := NEXT
3889:                      END;
3890:      1:            NEW(LPT3);
3891:                    WITH LPT3^ DO
3892:                      BEGIN NEXT := LPT1; CSLAB := LVAL.IVAL;
3893:                        CSSTART := IC
3894:                      END;
```

```
3895:                 IF LPT2 = NIL THEN FSTPTR := LPT3
3896:                 ELSE LPT2^.NEXT := LPT3
3897:               END
3898:             ELSE ERROR(147);
3899:           TEST := SY <> COMMA;
3900:           IF NOT TEST THEN INSYMBOL
3901:         UNTIL TEST;
3902:         IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
3903:         REPEAT STATEMENT(FSYS + [SEMICOLON])
3904:         UNTIL NOT (SY IN STATBEGSYS);
3905:         IF LPT3 <> NIL THEN
3906:           GENJMP(57(*UJP*),LADDR);
3907:         TEST := SY <> SEMICOLON;
3908:         IF NOT TEST THEN INSYMBOL
3909:       UNTIL TEST OR (SY = ENDSY);
3910:       PUTLABEL(LCIX);
3911:       IF FSTPTR <> NIL THEN
3912:         BEGIN LMAX := FSTPTR^.CSLAB;
3913:           LPT1 := FSTPTR; FSTPTR := NIL;
3914:           REPEAT LPT2 := LPT1^.NEXT; LPT1^.NEXT := FSTPTR;
3915:             FSTPTR := LPT1; LPT1 := LPT2
3916:           UNTIL LPT1 = NIL;
3917:           LMIN := FSTPTR^.CSLAB;
3918:               GEN0(44(*XJP*));
3919:               GENWORD(LMIN); GENWORD(LMAX);
3920:               NULSTMT := IC;
3921:               GENJMP(57(*UJP*),LADDR);
3922:               REPEAT
3923:                 WITH FSTPTR^ DO
3924:                   BEGIN
3925:                     WHILE CSLAB > LMIN DO
3926:                       BEGIN GENWORD(IC-NULSTMT); LMIN := LMIN + 1 END;
3927:                     GENWORD(IC-CSSTART);
3928:                     FSTPTR := NEXT; LMIN := LMIN + 1
3929:                   END
3930:               UNTIL FSTPTR = NIL;
3931:               PUTLABEL(LADDR)
3932:         END;
3933:         IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
3934:     END (*CASESTATEMENT*) ;
3935:
3936:     PROCEDURE REPEATSTATEMENT;
3937:       VAR LADDR: LBP;
3938:     BEGIN GENLABEL(LADDR); PUTLABEL(LADDR);
3939:       REPEAT
3940:         REPEAT STATEMENT(FSYS + [SEMICOLON,UNTILSY])
3941:         UNTIL NOT (SY IN STATBEGSYS);
3942:         TEST := SY <> SEMICOLON;
3943:         IF NOT TEST THEN INSYMBOL
3944:       UNTIL TEST;
3945:       IF SY = UNTILSY THEN
3946:         BEGIN INSYMBOL; EXPRESSION(FSYS); GENFJP(LADDR)
3947:         END
3948:       ELSE ERROR(53)
3949:     END (*REPEATSTATEMENT*) ;
3950:
3951:     PROCEDURE WHILESTATEMENT;
3952:       VAR LADDR, LCIX: LBP;
3953:     BEGIN GENLABEL(LADDR); PUTLABEL(LADDR);
3954:       EXPRESSION(FSYS + [DOSY]); GENLABEL(LCIX); GENFJP(LCIX);
3955:       IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
3956:       STATEMENT(FSYS); GENJMP(57(*UJP*),LADDR); PUTLABEL(LCIX)
3957:     END (*WHILESTATEMENT*) ;
3958:
3959:     PROCEDURE FORSTATEMENT;
3960:       VAR LATTR: ATTR; LSP: STP;  LSY: SYMBOL;
```

```
3961:           LCIX, LADDR: LBP;
3962:     BEGIN
3963:       IF SY = IDENT THEN
3964:         BEGIN SEARCHID([VARS],LCP);
3965:           WITH LCP^, LATTR DO
3966:             BEGIN TYPTR := IDTYPE; KIND := VARBL;
3967:               IF VKIND = ACTUAL THEN
3968:                 BEGIN ACCESS := DRCT; VLEVEL := VLEV;
3969:                   DPLMT := VADDR
3970:                 END
3971:               ELSE BEGIN ERROR(155); TYPTR := NIL END
3972:             END;
3973:           IF LATTR.TYPTR <> NIL THEN
3974:             IF (LATTR.TYPTR^.FORM > SUBRANGE)
3975:                OR COMPTYPES(REALPTR,LATTR.TYPTR) THEN
3976:               BEGIN ERROR(143); LATTR.TYPTR := NIL END;
3977:           INSYMBOL
3978:         END
3979:       ELSE
3980:         BEGIN ERROR(2); SKIP(FSYS + [BECOMES,TOSY,DOWNTOSY,DOSY])
3981:         END;
3982:       IF SY = BECOMES THEN
3983:         BEGIN INSYMBOL; EXPRESSION(FSYS + [TOSY,DOWNTOSY,DOSY]);
3984:           IF GATTR.TYPTR <> NIL THEN
3985:             IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(144)
3986:               ELSE
3987:                 IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3988:                   BEGIN LOAD;
3989:                     IF LATTR.TYPTR <> NIL THEN
3990:                       IF (LATTR.TYPTR^.FORM = SUBRANGE) AND RANGECHECK THEN
3991:                         BEGIN
3992:                           GENLDC(LATTR.TYPTR^.MIN.IVAL);
3993:                           GENLDC(LATTR.TYPTR^.MAX.IVAL);
3994:                           GEN0(8(*CHK*))
3995:                         END;
3996:                     STORE(LATTR)
3997:                   END
3998:                 ELSE ERROR(145)
3999:         END
4000:       ELSE
4001:         BEGIN ERROR(51); SKIP(FSYS + [TOSY,DOWNTOSY,DOSY]) END;
4002:       GENLABEL(LADDR);
4003:       IF SY IN [TOSY,DOWNTOSY] THEN
4004:         BEGIN LSY := SY; INSYMBOL; EXPRESSION(FSYS + [DOSY]);
4005:           IF GATTR.TYPTR <> NIL THEN
4006:             IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(144)
4007:             ELSE
4008:               IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
4009:                 BEGIN LOAD;
4010:                   IF LATTR.TYPTR <> NIL THEN
4011:                     IF (LATTR.TYPTR^.FORM = SUBRANGE) AND RANGECHECK THEN
4012:                       BEGIN
4013:                         GENLDC(LATTR.TYPTR^.MIN.IVAL);
4014:                         GENLDC(LATTR.TYPTR^.MAX.IVAL);
4015:                         GEN0(8(*CHK*))
4016:                       END;
4017:                   GEN2(56(*STR*),0,LC); PUTLABEL(LADDR);
4018:                   GATTR := LATTR; LOAD; GEN2(54(*LOD*),0,LC);
4019:                   LC := LC + INTSIZE;
4020:                   IF LC > LCMAX THEN LCMAX := LC;
4021:                   IF LSY = TOSY THEN GEN2(52(*LEQ*),0,INTSIZE)
4022:                   ELSE GEN2(48(*GEQ*),0,INTSIZE);
4023:                 END
4024:               ELSE ERROR(145)
4025:         END
4026:       ELSE BEGIN ERROR(55); SKIP(FSYS + [DOSY]) END;
```

```
4027:        GENLABEL(LCIX); GENJMP(33(*FJP*),LCIX);
4028:        IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
4029:        STATEMENT(FSYS);
4030:        GATTR := LATTR; LOAD; GENLDC(1);
4031:        IF LSY = TOSY THEN GEN0(2(*ADI*)) ELSE GEN0(21(*SBI*));
4032:        STORE(LATTR); GENJMP(57(*UJP*),LADDR); PUTLABEL(LCIX);
4033:        LC := LC - INTSIZE
4034:      END (*FORSTATEMENT*) ;
4035:
4036:
4037:      PROCEDURE WITHSTATEMENT;
4038:        VAR LCP: CTP; LCNT1,LCNT2: DISPRANGE;
4039:      BEGIN LCNT1 := 0; LCNT2 := 0;
4040:        REPEAT
4041:          IF SY = IDENT THEN
4042:            BEGIN SEARCHID([VARS,FIELD],LCP); INSYMBOL END
4043:          ELSE BEGIN ERROR(2); LCP := UVARPTR END;
4044:          SELECTOR(FSYS + [COMMA,DOSY],LCP);
4045:          IF GATTR.TYPTR <> NIL THEN
4046:            IF GATTR.TYPTR^.FORM = RECORDS THEN
4047:              IF TOP < DISPLIMIT THEN
4048:                BEGIN TOP := TOP + 1; LCNT1 := LCNT1 + 1;
4049:                  WITH DISPLAY[TOP] DO
4050:                    BEGIN FNAME := GATTR.TYPTR^.FSTFLD END;
4051:                  IF GATTR.ACCESS = DRCT THEN
4052:                    WITH DISPLAY[TOP] DO
4053:                      BEGIN OCCUR := CREC; CLEV := GATTR.VLEVEL;
4054:                        CDSPL := GATTR.DPLMT
4055:                      END
4056:                  ELSE
4057:                    BEGIN LOADADDRESS; GEN2(56(*STR*),0,LC);
4058:                      WITH DISPLAY[TOP] DO
4059:                        BEGIN OCCUR := VREC; VDSPL := LC END;
4060:                      LC := LC + PTRSIZE; LCNT2 := LCNT2 + PTRSIZE;
4061:                      IF LC > LCMAX THEN LCMAX := LC
4062:                    END
4063:                END
4064:              ELSE ERROR(250)
4065:            ELSE ERROR(140);
4066:          TEST := SY <> COMMA;
4067:          IF NOT TEST THEN INSYMBOL
4068:        UNTIL TEST;
4069:        IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
4070:        STATEMENT(FSYS);
4071:        TOP := TOP - LCNT1; LC := LC - LCNT2;
4072:      END (*WITHSTATEMENT*) ;
4073:
4074:    BEGIN (*STATEMENT*)
4075:      IF SY = INTCONST THEN (*LABEL*)
4076:        BEGIN TTOP := TOP;
4077:          WHILE DISPLAY[TTOP].OCCUR <> BLCK DO TTOP := TTOP-1;
4078:          LLP := DISPLAY[TTOP].FLABEL;
4079:          WHILE LLP <> NIL DO
4080:            WITH LLP^ DO
4081:              IF LABVAL = VAL.IVAL THEN
4082:                BEGIN
4083:                  IF CODELBP^.DEFINED THEN ERROR(165);
4084:                  PUTLABEL(CODELBP); GOTO 1
4085:                END
4086:              ELSE LLP := NEXTLAB;
4087:          ERROR(167);
4088:    1:    INSYMBOL;
4089:          IF SY = COLON THEN INSYMBOL ELSE ERROR(5)
4090:        END;
4091:      IF DEBUGGING THEN
4092:        BEGIN GEN1(85(*BPT*),SCREENDOTS+1); BPTONLINE := TRUE END;
```

```
4093:      IF NOT (SY IN FSYS + [IDENT]) THEN
4094:        BEGIN ERROR(6); SKIP(FSYS) END;
4095:      IF SY IN STATBEGSYS + [IDENT] THEN
4096:        BEGIN MARK(HEAP); (*FOR LABEL CLEANUP*)
4097:          CASE SY OF
4098:            IDENT:    BEGIN SEARCHID([VARS,FIELD,FUNC,PROC],LCP);
4099:                        INSYMBOL;
4100:                        IF LCP^.KLASS = PROC THEN CALL(FSYS,LCP)
4101:                        ELSE ASSIGNMENT(LCP)
4102:                      END;
4103:            BEGINSY:  BEGIN INSYMBOL; COMPOUNDSTATEMENT END;
4104:            GOTOSY:   BEGIN INSYMBOL; GOTOSTATEMENT END;
4105:            IFSY:     BEGIN INSYMBOL; IFSTATEMENT END;
4106:            CASESY:   BEGIN INSYMBOL; CASESTATEMENT END;
4107:            WHILESY:  BEGIN INSYMBOL; WHILESTATEMENT END;
4108:            REPEATSY: BEGIN INSYMBOL; REPEATSTATEMENT END;
4109:            FORSY:    BEGIN INSYMBOL; FORSTATEMENT END;
4110:            WITHSY:   BEGIN INSYMBOL; WITHSTATEMENT END
4111:          END;
4112:          RELEASE(HEAP);
4113:          IF IC + 100 > MAXCODE THEN
4114:            BEGIN ERROR(253); IC := 0 END;
4115:          IF NOT (SY IN [SEMICOLON,ENDSY,ELSESY,UNTILSY]) THEN
4116:            BEGIN ERROR(6); SKIP(FSYS) END
4117:        END
4118:    END (*STATEMENT*) ;
4119:   (*    COPYRIGHT (C) 1978, REGENTS OF THE       *)
4120:  (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
4121:
4122:    PROCEDURE BLOCK(FSYS: SETOFSYS; FSY: SYMBOL; FPROCP: CTP);
4123:      VAR LSY: SYMBOL;
4124:
4125:      PROCEDURE LABELDECLARATION;
4126:        VAR LLP: LABELP; REDEF: BOOLEAN;
4127:      BEGIN
4128:        REPEAT
4129:          IF SY = INTCONST THEN
4130:            WITH DISPLAY[TOP] DO
4131:              BEGIN LLP := FLABEL; REDEF := FALSE;
4132:                WHILE (LLP <> NIL) AND NOT REDEF DO
4133:                  IF LLP^.LABVAL <> VAL.IVAL THEN
4134:                    LLP := LLP^.NEXTLAB
4135:                  ELSE BEGIN REDEF := TRUE; ERROR(166) END;
4136:                IF NOT REDEF THEN
4137:                  BEGIN NEW(LLP);
4138:                    WITH LLP^ DO
4139:                      BEGIN LABVAL := VAL.IVAL;
4140:                        CODELBP := NIL; NEXTLAB := FLABEL
4141:                      END;
4142:                    FLABEL := LLP
4143:                  END;
4144:                INSYMBOL
4145:              END
4146:          ELSE ERROR(15);
4147:          IF NOT ( SY IN FSYS + [COMMA, SEMICOLON] ) THEN
4148:            BEGIN ERROR(6); SKIP(FSYS+[COMMA,SEMICOLON]) END;
4149:          TEST := SY <> COMMA;
4150:          IF NOT TEST THEN INSYMBOL
4151:        UNTIL TEST;
4152:        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
4153:      END (* LABELDECLARATION *) ;
4154:
4155:      PROCEDURE CONSTDECLARATION;
4156:        VAR LCP: CTP; LSP: STP; LVALU: VALU;
4157:      BEGIN
4158:        IF SY <> IDENT THEN
```

```
4159:           BEGIN ERROR(2); SKIP(FSYS + [IDENT]) END;
4160:         WHILE SY = IDENT DO
4161:           BEGIN NEW(LCP,KONST);
4162:             WITH LCP^ DO
4163:               BEGIN NAME := ID; IDTYPE := NIL;
4164:                 NEXT := NIL; KLASS := KONST
4165:               END;
4166:             INSYMBOL;
4167:             IF (SY = RELOP) AND (OP = EQOP) THEN INSYMBOL ELSE ERROR(16);
4168:             CONSTANT(FSYS + [SEMICOLON],LSP,LVALU);
4169:             ENTERID(LCP);
4170:             LCP^.IDTYPE := LSP; LCP^.VALUES := LVALU;
4171:             IF SY = SEMICOLON THEN
4172:               BEGIN INSYMBOL;
4173:                 IF NOT (SY IN FSYS + [IDENT]) THEN
4174:                   BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
4175:               END
4176:             ELSE ERROR(14)
4177:           END
4178:       END (*CONSTDECLARATION*) ;
4179:
4180:       PROCEDURE TYPEDECLARATION;
4181:         VAR LCP,LCP1,LCP2: CTP; LSP: STP; LSIZE: ADDRRANGE;
4182:       BEGIN
4183:         IF SY <> IDENT THEN
4184:           BEGIN ERROR(2); SKIP(FSYS + [IDENT]) END;
4185:         WHILE SY = IDENT DO
4186:           BEGIN NEW(LCP,TYPES);
4187:             WITH LCP^ DO
4188:               BEGIN NAME := ID; IDTYPE := NIL; KLASS := TYPES END;
4189:             INSYMBOL;
4190:             IF (SY = RELOP) AND (OP = EQOP) THEN INSYMBOL ELSE ERROR(16);
4191:             TYP(FSYS + [SEMICOLON],LSP,LSIZE);
4192:             ENTERID(LCP);
4193:             LCP^.IDTYPE := LSP;
4194:             LCP1 := FWPTR;
4195:             WHILE LCP1 <> NIL DO
4196:               BEGIN
4197:                 IF LCP1^.NAME = LCP^.NAME THEN
4198:                   BEGIN
4199:                     LCP1^.IDTYPE^.ELTYPE := LCP^.IDTYPE;
4200:                     IF LCP1 <> FWPTR THEN
4201:                       LCP2^.NEXT := LCP1^.NEXT
4202:                     ELSE FWPTR := LCP1^.NEXT;
4203:                   END;
4204:                 LCP2 := LCP1; LCP1 := LCP1^.NEXT
4205:               END;
4206:             IF SY = SEMICOLON THEN
4207:               BEGIN INSYMBOL;
4208:                 IF NOT (SY IN FSYS + [IDENT]) THEN
4209:                   BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
4210:               END
4211:             ELSE ERROR(14)
4212:           END;
4213:         IF FWPTR <> NIL THEN
4214:           BEGIN ERROR(117); FWPTR := NIL END
4215:       END (*TYPEDECLARATION*) ;
4216:
4217:       PROCEDURE VARDECLARATION;
4218:         VAR LCP,NXT,IDLIST: CTP; LSP: STP; LSIZE: ADDRRANGE;
4219:       BEGIN NXT := NIL;
4220:         REPEAT
4221:           REPEAT
4222:             IF SY = IDENT THEN
4223:               BEGIN NEW(LCP,VARS);
4224:                 WITH LCP^ DO
```

```
4225:                BEGIN NAME := ID; NEXT := NXT; KLASS := VARS;
4226:                   IDTYPE := NIL; VKIND := ACTUAL; VLEV := LEVEL
4227:                 END;
4228:               ENTERID(LCP);
4229:               NXT := LCP;
4230:               INSYMBOL;
4231:             END
4232:           ELSE ERROR(2);
4233:           IF NOT (SY IN FSYS + [COMMA,COLON] + TYPEDELS) THEN
4234:             BEGIN ERROR(6); SKIP(FSYS+[COMMA,COLON,SEMICOLON]+TYPEDELS) END;
4235:           TEST := SY <> COMMA;
4236:           IF NOT TEST THEN INSYMBOL
4237:         UNTIL TEST;
4238:         IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
4239:         IDLIST := NXT;
4240:         TYP(FSYS + [SEMICOLON] + TYPEDELS,LSP,LSIZE);
4241:         WHILE NXT <> NIL DO
4242:           WITH  NXT^ DO
4243:             BEGIN IDTYPE := LSP; VADDR := LC;
4244:               LC := LC + LSIZE; NXT := NEXT;
4245:               IF NEXT = NIL THEN
4246:                 IF LSP <> NIL THEN
4247:                   IF LSP^.FORM = FILES THEN
4248:                       BEGIN (*PUT IDLIST INTO LOCAL FILE LIST*)
4249:                         NEXT := DISPLAY[TOP].FFILE;
4250:                         DISPLAY[TOP].FFILE := IDLIST
4251:                       END
4252:             END;
4253:         IF SY = SEMICOLON THEN
4254:           BEGIN INSYMBOL;
4255:             IF NOT (SY IN FSYS + [IDENT]) THEN
4256:               BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
4257:           END
4258:         ELSE ERROR(14)
4259:       UNTIL (SY <> IDENT) AND NOT (SY IN TYPEDELS);
4260:     IF FWPTR <> NIL THEN
4261:         BEGIN ERROR(117); FWPTR := NIL END
4262:     END (*VARDECLARATION*) ;
4263:
4264:     PROCEDURE PROCDECLARATION(FSY: SYMBOL; SEGDEC: BOOLEAN);
4265:       VAR OLDLEV: 0..MAXLEVEL; LSY: SYMBOL; LCP,LCP1: CTP; LSP: STP;
4266:           FORW: BOOLEAN; OLDTOP: DISPRANGE; OLDPROC: PROCRANGE;
4267:           LLC,LCM: ADDRRANGE;  MARKP: ^INTEGER;
4268:
4269:       PROCEDURE PARAMETERLIST(FSY: SETOFSYS; VAR FPAR: CTP; FCP: CTP);
4270:         VAR LCP,LCP1,LCP2,LCP3: CTP; LSP: STP; LKIND: IDKIND;
4271:           LLC,LEN : ADDRRANGE; COUNT : INTEGER;
4272:       BEGIN LCP1 := NIL; LLC := LC;
4273:         IF NOT (SY IN FSY + [LPARENT]) THEN
4274:           BEGIN ERROR(7); SKIP(FSYS + FSY + [LPARENT]) END;
4275:         IF SY = LPARENT THEN
4276:           BEGIN IF FORW THEN ERROR(119);
4277:             INSYMBOL;
4278:             IF NOT (SY IN [IDENT,VARSY]) THEN
4279:               BEGIN ERROR(7); SKIP(FSYS + [IDENT,RPARENT]) END;
4280:             WHILE SY IN [IDENT,VARSY] DO
4281:               BEGIN
4282:                 IF SY = VARSY THEN
4283:                   BEGIN LKIND := FORMAL; INSYMBOL END
4284:                 ELSE LKIND := ACTUAL;
4285:                 LCP2 := NIL;
4286:                 COUNT := 0;
4287:                 REPEAT
4288:                   IF SY <> IDENT THEN ERROR(2)
4289:                   ELSE
4290:                     BEGIN NEW(LCP,VARS);
```

```
4291:                       WITH LCP^ DO
4292:                         BEGIN NAME := ID; IDTYPE := NIL;
4293:                           VKIND := LKIND; NEXT := LCP2;
4294:                           KLASS := VARS; VLEV := LEVEL
4295:                         END;
4296:                       ENTERID(LCP);
4297:                       LCP2 := LCP; COUNT := COUNT + 1;
4298:                       INSYMBOL
4299:                     END;
4300:                   IF NOT (SY IN FSYS + [COMMA,COLON]) THEN
4301:                     BEGIN ERROR(7);
4302:                       SKIP(FSYS + [COMMA,SEMICOLON,RPARENT,COLON])
4303:                     END;
4304:                   TEST := SY <> COMMA;
4305:                   IF NOT TEST THEN INSYMBOL
4306:                 UNTIL TEST;
4307:                 IF SY = COLON THEN
4308:                   BEGIN INSYMBOL;
4309:                     IF SY = IDENT THEN
4310:                       BEGIN
4311:                         SEARCHID([TYPES],LCP);
4312:                         LSP := LCP^.IDTYPE;
4313:                         LCP3 := LCP2;
4314:                         LEN := PTRSIZE;
4315:                         IF LSP <> NIL THEN
4316:                           IF LKIND = ACTUAL THEN
4317:                             IF LSP^.FORM = FILES THEN ERROR(121)
4318:                             ELSE
4319:                               IF LSP^.FORM <= POWER THEN LEN := LSP^.SIZE;
4320:                         LC := LC + COUNT * LEN;
4321:                         WHILE LCP2 <> NIL DO
4322:                           BEGIN LCP := LCP2;
4323:                             WITH LCP2^ DO
4324:                               BEGIN IDTYPE := LSP;
4325:                                 LCP2 := NEXT
4326:                               END
4327:                           END;
4328:                         LCP^.NEXT := LCP1; LCP1 := LCP3;
4329:                         INSYMBOL
4330:                       END
4331:                     ELSE ERROR(2);
4332:                     IF NOT (SY IN FSYS + [SEMICOLON,RPARENT]) THEN
4333:                       BEGIN ERROR(7); SKIP(FSYS + [SEMICOLON,RPARENT]) END;
4334:                   END
4335:                 ELSE ERROR(5);
4336:                 IF SY = SEMICOLON THEN
4337:                   BEGIN INSYMBOL;
4338:                     IF NOT (SY IN FSYS + [IDENT,VARSY]) THEN
4339:                       BEGIN ERROR(7); SKIP(FSYS + [IDENT,RPARENT]) END
4340:                   END
4341:               END (*WHILE*) ;
4342:             IF SY = RPARENT THEN
4343:               BEGIN INSYMBOL;
4344:                 IF NOT (SY IN FSY + FSYS) THEN
4345:                   BEGIN ERROR(6); SKIP(FSY + FSYS) END
4346:               END
4347:             ELSE ERROR(4);
4348:             FCP^.LOCALLC := LC; LCP3 := NIL;
4349:             WHILE LCP1 <> NIL DO
4350:               WITH LCP1^ DO
4351:                 BEGIN LCP2 := NEXT; NEXT := LCP3;
4352:                   IF (KLASS = VARS) AND (IDTYPE <> NIL) THEN
4353:                     IF (IDTYPE^.FORM <= POWER) OR (VKIND = FORMAL) THEN
4354:                       BEGIN VADDR := LLC;
4355:                         IF VKIND = FORMAL THEN LLC := LLC + PTRSIZE
4356:                         ELSE LLC := LLC + IDTYPE^.SIZE
```

```
4357:                       END
4358:                     ELSE
4359:                       BEGIN VADDR := LC;
4360:                         LC := LC + IDTYPE^.SIZE;
4361:                         LLC := LLC + PTRSIZE
4362:                       END;
4363:                   LCP3 := LCP1; LCP1 := LCP2
4364:                 END;
4365:             FPAR := LCP3
4366:           END
4367:             ELSE FPAR := NIL
4368:     END (*PARAMETERLIST*) ;
4369:
4370:     BEGIN (*PROCDECLARATION*)
4371:       LLC := LC; LC := LCAFTERMARKSTACK;
4372:       IF FSY = FUNCSY THEN LC := LC + REALSIZE;
4373:       LINEINFO := LC; DP := TRUE;
4374:       IF SY = IDENT THEN
4375:         BEGIN SEARCHSECTION(DISPLAY[TOP].FNAME,LCP);
4376:           IF LCP <> NIL THEN
4377:            BEGIN
4378:             IF LCP^.KLASS = PROC THEN
4379:               FORW := LCP^.FORWDECL AND (FSY = PROCSY)
4380:                       AND (LCP^.PFKIND = ACTUAL)
4381:             ELSE
4382:               IF LCP^.KLASS = FUNC THEN
4383:                 FORW := LCP^.FORWDECL AND (FSY = FUNCSY)
4384:                         AND (LCP^.PFKIND = ACTUAL)
4385:               ELSE FORW := FALSE;
4386:             IF NOT FORW THEN ERROR(160)
4387:            END
4388:           ELSE FORW := FALSE;
4389:           IF NOT FORW THEN
4390:             BEGIN
4391:               IF FSY = PROCSY THEN NEW(LCP,PROC,DECLARED,ACTUAL)
4392:               ELSE NEW(LCP,FUNC,DECLARED,ACTUAL);
4393:               WITH LCP^ DO
4394:                 BEGIN NAME := ID; IDTYPE := NIL; LOCALLC := LC;
4395:                   PFDECKIND := DECLARED; PFKIND := ACTUAL;
4396:                   INSCOPE := FALSE; PFLEV := LEVEL;
4397:                   PFNAME := NEXTPROC; PFSEG := SEG;
4398:                   IF SEGDEC THEN NEXTSEG := NEXTSEG+1;
4399:                   IF NEXTPROC = MAXPROCNUM THEN ERROR(251)
4400:                   ELSE NEXTPROC := NEXTPROC + 1;
4401:                   IF FSY = PROCSY THEN KLASS := PROC
4402:                   ELSE KLASS := FUNC
4403:                 END;
4404:               ENTERID(LCP)
4405:             END
4406:           ELSE
4407:             BEGIN LCP1 := LCP^.NEXT;
4408:               WHILE LCP1 <> NIL DO
4409:                 BEGIN
4410:                   WITH LCP1^ DO
4411:                     IF KLASS = VARS THEN
4412:                       IF IDTYPE <> NIL THEN
4413:                         BEGIN
4414:                           IF VKIND = FORMAL THEN LCM := VADDR + PTRSIZE
4415:                           ELSE LCM := VADDR + IDTYPE^.SIZE;
4416:                           IF LCM > LC THEN LC := LCM
4417:                         END;
4418:                   LCP1 := LCP1^.NEXT
4419:                 END;
4420:               IF SEG <> LCP^.PFSEG THEN
4421:                 BEGIN
4422:                   SEG := LCP^.PFSEG; NEXTPROC := 2;
```

```
4423:                   IF NOT SEGDEC THEN ERROR(399)
4424:                 END
4425:             END;
4426:           INSYMBOL
4427:         END
4428:       ELSE
4429:         BEGIN ERROR(2); LCP := UPRCPTR END;
4430:       OLDLEV := LEVEL; OLDTOP := TOP; OLDPROC := CURPROC;
4431:       CURPROC := LCP^.PFNAME;
4432:       IF LEVEL < MAXLEVEL THEN LEVEL := LEVEL + 1 ELSE ERROR(251);
4433:       IF TOP < DISPLIMIT THEN
4434:         BEGIN TOP := TOP + 1;
4435:           WITH DISPLAY[TOP] DO
4436:             BEGIN
4437:               IF FORW THEN FNAME := LCP^.NEXT
4438:               ELSE FNAME := NIL;
4439:               FLABEL := NIL; FFILE := NIL; OCCUR := BLCK
4440:             END
4441:         END
4442:       ELSE ERROR(250);
4443:       IF FSY = PROCSY THEN
4444:         BEGIN PARAMETERLIST([SEMICOLON],LCP1,LCP);
4445:           IF NOT FORW THEN LCP^.NEXT := LCP1
4446:         END
4447:       ELSE
4448:         BEGIN PARAMETERLIST([SEMICOLON,COLON],LCP1,LCP);
4449:           IF NOT FORW THEN LCP^.NEXT := LCP1;
4450:           IF SY = COLON THEN
4451:             BEGIN INSYMBOL;
4452:               IF SY = IDENT THEN
4453:                 BEGIN IF FORW THEN ERROR(122);
4454:                   SEARCHID([TYPES],LCP1);
4455:                   LSP := LCP1^.IDTYPE;
4456:                   LCP^.IDTYPE := LSP;
4457:                   IF LSP <> NIL THEN
4458:                     IF NOT (LSP^.FORM IN [SCALAR,SUBRANGE,POINTER]) THEN
4459:                       BEGIN ERROR(120); LCP^.IDTYPE := NIL END;
4460:                   INSYMBOL
4461:                 END
4462:               ELSE BEGIN ERROR(2); SKIP(FSYS + [SEMICOLON]) END
4463:             END
4464:           ELSE
4465:             IF NOT FORW THEN ERROR(123)
4466:         END;
4467:       IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
4468:       IF SY = FORWARDSY THEN
4469:         BEGIN
4470:           IF FORW THEN ERROR(161)
4471:           ELSE LCP^.FORWDECL := TRUE;
4472:           INSYMBOL;
4473:           IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
4474:           IF NOT (SY IN FSYS) THEN
4475:             BEGIN ERROR(6); SKIP(FSYS) END
4476:         END
4477:       ELSE
4478:         BEGIN MARK(MARKP);
4479:           WITH LCP^ DO
4480:             BEGIN FORWDECL := FALSE; INSCOPE := TRUE END;
4481:           REPEAT BLOCK(FSYS,SEMICOLON,LCP);
4482:             RELEASE(MARKP);
4483:             IF SY = SEMICOLON THEN
4484:               BEGIN INSYMBOL;
4485:                 IF NOT (SY IN [BEGINSY,PROCSY,FUNCSY,PROGSY]) THEN
4486:                   BEGIN ERROR(6); SKIP(FSYS) END
4487:               END
4488:             ELSE ERROR(14)
```

```
4489:            UNTIL SY IN [BEGINSY,PROCSY,FUNCSY,PROGSY];
4490:            LCP^.INSCOPE := FALSE
4491:          END;
4492:        LEVEL := OLDLEV; TOP := OLDTOP; LC := LLC; CURPROC := OLDPROC
4493:      END (*PROCDECLARATION*) ;
4494:
4495:      PROCEDURE SEGDECLARATION;
4496:        VAR LSY: SYMBOL; OLDPROC: PROCRANGE; OLDSEG: SEGRANGE;
4497:      BEGIN
4498:        IF CODEINSEG THEN
4499:          BEGIN ERROR(399); SEGINX := 0; CURBYTE := 0 END;
4500:        OLDSEG := SEG; SEG := NEXTSEG; OLDPROC := NEXTPROC;
4501:        IF NEXTSEG > MAXSEG THEN ERROR(250);
4502:        NEXTPROC := 1; LSY := SY;
4503:        IF SY IN [PROCSY,FUNCSY] THEN INSYMBOL
4504:        ELSE
4505:          BEGIN ERROR(399); LSY := PROCSY END;
4506:        IF SY = IDENT THEN SEGTABLE[SEG].SEGNAME := ID;
4507:        PROCDECLARATION(LSY,TRUE);
4508:        IF CODEINSEG THEN FINISHSEG;
4509:        NEXTPROC := OLDPROC; SEG := OLDSEG
4510:      END (*SEGDECLARATION*) ;
4511:
4512:      PROCEDURE BODY(FSYS: SETOFSYS);
4513:        VAR LLC1,EXITIC: ADDRRANGE; LCP,LLCP: CTP; LOP: OPRANGE;
4514:            LLP: LABELP; LMIN,LMAX: INTEGER; JTINX: JTABRANGE;
4515:
4516:      BEGIN NEXTJTAB := 1;
4517:        IF NOISY THEN
4518:          BEGIN WRITELN(OUTPUT);
4519:            IF FPROCP <> NIL THEN
4520:              WRITE(OUTPUT,FPROCP^.NAME);
4521:            WRITELN(OUTPUT);
4522:            WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
4523:          END;
4524:        IF FPROCP <> NIL THEN
4525:          BEGIN
4526:            LLC1 := FPROCP^.LOCALLC; LCP := FPROCP^.NEXT;
4527:            WHILE LCP <> NIL DO
4528:              WITH LCP^ DO
4529:                BEGIN
4530:                  IF KLASS = VARS THEN
4531:                    IF IDTYPE <> NIL THEN
4532:                      IF (VKIND = ACTUAL) AND (IDTYPE^.FORM > POWER) THEN
4533:                        BEGIN LLC1 := LLC1 - PTRSIZE;
4534:                          GEN2(50(*LDA*),0,VADDR);
4535:                          GEN2(54(*LOD*),0,LLC1);
4536:                          IF PAOFCHAR(IDTYPE) THEN
4537:                            WITH IDTYPE^ DO
4538:                              IF AISSTRNG THEN GEN1(42(*SAS*),MAXLENG)
4539:                              ELSE
4540:                                IF INXTYPE <> NIL THEN
4541:                                  BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
4542:                                    GEN1(41(*MVB*),LMAX - LMIN + 1)
4543:                                  END
4544:                                ELSE
4545:                          ELSE GEN1(40(*MOV*),IDTYPE^.SIZE)
4546:                        END
4547:                      ELSE
4548:                        IF VKIND = FORMAL THEN LLC1 := LLC1 - PTRSIZE
4549:                        ELSE LLC1 := LLC1 - IDTYPE^.SIZE;
4550:                  LCP := NEXT
4551:                END;
4552:          END;
4553:        STARTDOTS := SCREENDOTS;
4554:        LCMAX := LC;
```

```
4555:       LLP := DISPLAY[TOP].FLABEL;
4556:       WHILE LLP <> NIL DO
4557:         BEGIN GENLABEL(LLP^.CODELBP);
4558:           LLP := LLP^.NEXTLAB
4559:         END;
4560:       LCP := DISPLAY[TOP].FFILE;
4561:       WHILE LCP <> NIL DO
4562:         WITH LCP^,IDTYPE^ DO
4563:           BEGIN
4564:             GEN2(50(*LDA*),0,VADDR);
4565:             GEN2(50(*LDA*),0,VADDR+FILESIZE);
4566:             IF FILTYPE = NIL THEN GENLDC(-1)
4567:             ELSE
4568:               IF IDTYPE = INTRACTVPTR THEN GENLDC(0)
4569:               ELSE
4570:                 IF FILTYPE = CHARPTR THEN GENLDC(-2)
4571:                 ELSE GENLDC(FILTYPE^.SIZE);
4572:             GEN2(77(*CXP*),0(*SYS*),3(*FINIT*));
4573:             LCP := NEXT
4574:           END;
4575:       IF (LEVEL = 1) AND NOT SYSCOMP THEN
4576:         GEN1(85(*BPT*),SCREENDOTS+1);
4577:       REPEAT
4578:         REPEAT STATEMENT(FSYS + [SEMICOLON,ENDSY])
4579:         UNTIL NOT (SY IN STATBEGSYS);
4580:         TEST := SY <> SEMICOLON;
4581:         IF NOT TEST THEN INSYMBOL
4582:       UNTIL TEST;
4583:       IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13);
4584:       EXITIC := IC;
4585:       LCP := DISPLAY[TOP].FFILE;
4586:       WHILE LCP <> NIL DO
4587:         WITH LCP^ DO
4588:           BEGIN
4589:             GEN2(50(*LDA*),0,VADDR);
4590:             GENLDC(0); GEN2(77(*CXP*),0(*SYS*),6(*FCLOSE*));
4591:             LCP := NEXT
4592:           END;
4593:       IF FPROCP = NIL THEN GEN0(86(*XIT*))
4594:       ELSE
4595:         BEGIN
4596:           IF FPROCP^.PFLEV = 0 THEN LOP := 65(*RBP*)
4597:           ELSE LOP := 45(*RNP*);
4598:           IF FPROCP^.IDTYPE = NIL THEN GEN1(LOP,0)
4599:           ELSE GEN1(LOP,FPROCP^.IDTYPE^.SIZE)
4600:         END;
4601:       LLP := DISPLAY[TOP].FLABEL;  (* CHECK UNDEFINED LABELS *)
4602:       WHILE LLP <> NIL DO
4603:         WITH LLP^,CODELBP^ DO
4604:           BEGIN
4605:             IF NOT DEFINED THEN
4606:               IF REFLIST <> MAXADDR THEN ERROR(168);
4607:             LLP := NEXTLAB
4608:           END;
4609:       JTINX := NEXTJTAB - 1;
4610:       IF ODD(IC) THEN IC := IC + 1;
4611:       WHILE JTINX > 0 DO
4612:         BEGIN GENWORD(IC-JTAB[JTINX]); JTINX := JTINX-1 END;
4613:       IF FPROCP = NIL THEN
4614:         BEGIN GENWORD((LCMAX-LCAFTERMARKSTACK)*2); GENWORD(0) END
4615:       ELSE
4616:         WITH FPROCP^ DO
4617:           BEGIN GENWORD((LCMAX-LOCALLC)*2);
4618:             GENWORD((LOCALLC-LCAFTERMARKSTACK)*2)
4619:           END;
4620:       GENWORD(IC-EXITIC); GENWORD(IC);
```

```
4621:        GENBYTE(CURPROC); GENBYTE(LEVEL-1);
4622:        IF NOT CODEINSEG THEN
4623:          BEGIN CODEINSEG := TRUE;
4624:            SEGTABLE[SEG].DISKADDR := CURBLK
4625:          END;
4626:        WRITECODE(FALSE);
4627:        SEGINX := SEGINX + IC;
4628:        PROCTABLE[CURPROC] := SEGINX - 2
4629:      END (*BODY*) ;
4630:
4631:    PROCEDURE FINDFORW(FCP: CTP);
4632:    BEGIN
4633:      IF FCP <> NIL THEN
4634:        WITH FCP^ DO
4635:          BEGIN
4636:            IF KLASS IN [PROC,FUNC] THEN
4637:              IF PFDECKIND = DECLARED THEN
4638:                IF PFKIND = ACTUAL THEN
4639:                  IF FORWDECL THEN
4640:                    BEGIN
4641:                      USERINFO.ERRNUM := 117; WRITELN(OUTPUT);
4642:                      WRITE(OUTPUT,NAME,' UNDEFINED')
4643:                    END;
4644:            FINDFORW(RLINK); FINDFORW(LLINK)
4645:          END
4646:    END (*FINDFORW*) ;
4647:
4648:    BEGIN (*BLOCK*)
4649:      REPEAT
4650:        IF SY = LABELSY THEN
4651:          BEGIN INSYMBOL; LABELDECLARATION END;
4652:        IF SY = CONSTSY THEN
4653:          BEGIN INSYMBOL; CONSTDECLARATION END;
4654:        IF SY = TYPESY THEN
4655:          BEGIN INSYMBOL; TYPEDECLARATION END;
4656:        IF SY = VARSY THEN
4657:          BEGIN INSYMBOL; VARDECLARATION END;
4658:        WHILE SY IN [PROCSY,FUNCSY,PROGSY] DO
4659:          BEGIN LSY := SY; INSYMBOL;
4660:            IF LSY = PROGSY THEN SEGDECLARATION
4661:            ELSE PROCDECLARATION(LSY,FALSE)
4662:          END;
4663:        IF SY <> BEGINSY THEN
4664:          IF NOT (INCLUDING AND
4665:              (SY IN [LABELSY,CONSTSY,TYPESY,VARSY,PROCSY,FUNCSY,PROGSY])) THEN
4666:            BEGIN ERROR(18); SKIP(FSYS) END
4667:      UNTIL SY IN STATBEGSYS;
4668:      DP := FALSE; IC := 0; LINEINFO := 0;
4669:      IF SY = BEGINSY THEN INSYMBOL ELSE ERROR(17);
4670:      IF NOT SYSCOMP THEN FINDFORW(DISPLAY[TOP].FNAME);
4671:      REPEAT BODY(FSYS + [CASESY]);
4672:        IF SY <> FSY THEN
4673:          BEGIN ERROR(6); SKIP(FSYS + [FSY]) END
4674:      UNTIL (SY = FSY) OR (SY IN BLOCKBEGSYS);
4675:    END (*BLOCK*) ;
4676:
4677:  BEGIN (*BLOCKCOMPILE*)
4678:    IF NOISY THEN
4679:      UNITWRITE(3,DUMMYVAR[-1600],35); (* TURN ON DISPLAY OF STACK AND HEAP *)
4680:    TIME(LGTH,LOWTIME); (* <<<< SMF 2-25-78 COMPINIT CALLED FROM PASCALCOMPILER*)
4681:    BLOCK(BLOCKBEGSYS+STATBEGSYS-[CASESY],PERIOD,OUTERBLOCK);
4682:    IF SY <> PERIOD THEN ERROR(21);
4683:    IF LIST THEN
4684:      BEGIN SCREENDOTS := SCREENDOTS+1;
4685:        SYMBUFP^[SYMCURSOR] := CHR(EOL);
4686:        SYMCURSOR := SYMCURSOR+1;
```

```
4687:        PRINTLINE
4688:      END;
4689:    FINISHSEG; USERINFO.ERRBLK := 0;
4690:    TIME(LGTH,STARTDOTS); LOWTIME := STARTDOTS-LOWTIME;
4691:    UNITWRITE(3,IC,7); CLOSE(LP,LOCK);
4692:    IF NOISY THEN WRITELN(OUTPUT);
4693:    WRITE(OUTPUT,SCREENDOTS,' LINES');
4694:    IF LOWTIME > 0 THEN
4695:      WRITE(OUTPUT,', ',(LOWTIME+30) DIV 60,' SECS, ',
4696:          ROUND((3600/LOWTIME)*SCREENDOTS),' LINES/MIN');
4697:    IC := 0;
4698:    FOR SEG := 0 TO MAXSEG DO
4699:      WITH SEGTABLE[SEG] DO
4700:        BEGIN GENWORD(DISKADDR); GENWORD(CODELENG) END;
4701:    FOR SEG := 0 TO MAXSEG DO
4702:      WITH SEGTABLE[SEG] DO
4703:        FOR LGTH := 1 TO 8 DO
4704:          GENBYTE(ORD(SEGNAME[LGTH]));
4705:    CURBLK := 0; CURBYTE := 0; WRITECODE(TRUE)
4706:  END (*BLOCKCOMPILE*) ;
4707:
4708:  BEGIN (* PASCALCOMPILER *)
4709:    COMPINIT;
4710:    BLOCKCOMPILE;
4711:  END;
4712:
4713:  BEGIN (* SYSTEM *)
4714:  END.
4715:
4716:

THAT'S ALL FOLKS!      LINES: 4716   CHARACTERS: 164764
```