```
          IDENT    POSYS,101B,POSYS.1
          ABS
          SPACE    10
*********************************************************
*                                                       *
*                                                       *
*                  ***************                      *
*                  *             *                      *
*                  *  P O S Y S  *                      *
*                  *             *                      *
*                  ***************                      *
*                                                       *
*                    (NOV. 1972)                        *
*                                                       *
*                                                       *
*          AUTHORS:  E. MARMIER                         *
*                    R. SCHILD                          *
*                    K. JENSEN                          *
*                                                       *
*                    FACHGRUPPE COMPUTERWISSENSCHAFTEN  *
*                    EIDG. TECHNISCHE HOCHSCHULE        *
*                    CLAUSIUSSTR. 55                    *
*           CH-8006  ZUERICH                            *
*                    -------                            *
*                    SWITZERLAND                        *
*                                                       *
*                                                       *
*********************************************************
          EJECT
  TRANSFER VECTORS:
  ---------------------------------------------------------------

          ORG      101B
    TRANSFER VECTOR FOR IOPAK ROUTINES:
          EQ       GETC
          EQ       PUTB
          EQ       PUTC
          EQ       OPEN
          EQ       CLOSE
          EQ       RESET
          EQ       WEOR
          EQ       READ
          EQ       WRITEOCT
          EQ       TEXT
          EQ       GETB
          EQ       INTGR
          EQ       EREAL
          EQ       FREAL
          EQ       ALF
          EQ       CHAR
          EQ       BOOL
          EQ       REWRITE

          ORG      1328
      TRANSFER VECTOR FOR EXTERNAL ROUTINES:
    EXTVECT  EQU       *
    SINCOS   VFD       60/EXP           SIN
             DATA      0                COS
    EXP      VFD       60/LN            EXP
    LN       VFD       60/SQRT          LN
    SQRT     VFD       60/ARCTAN        SQRT
    ARCTAN   VFD       60/NULL          ARCTAN
    NULL     DATA      0

    PMD      VFD       60/NULL1
    NULL1    DATA      0
```

```
  ABNORMAL EXIT TRANSFERS:
          ORG     1/0B
ABN.EX    MACRO   MSG
          EQ      ABN.EX1
          ENDM

          ABN.EX  ASSMSG          FOR $A+ COMPILER OPTION
          ABN.EX  INDXMSG         FOR $X+ COMPILER OPTION
          ABN.EX  DIVMSG          FOR $D+ COMPILER OPTION
          BSS     1
          LX7     42
          SB5     X7
          LX7     18
          ABN.EX  OVFLMSG         FOR $O+ COMPILER OPTION
          BSS     1
  TRANSFER WORD FOR ABNORMAL EXITS OUT OF EXTERNAL ROUTINES:
          BX7     X1
          EQ      ABN.EX1         X0 ALREADY CONTAINS THE MESSAGE ADDRESS
          EJECT
  EFETS AND CHARACTER BUFFERS FOR INPUT AND OUTPUT,
  FETS FOR PASCAL, PASCLGO, THE KEEP FILE, AND PMDFILE
  (THE BUFFERS ARE LOCATED AT THE END OF POSYS):
  ------------------------------------------------------------

BIGBUF    EQU     5000B
BUFLGTH   EQU     1001B

 EFET FORMAT (FET - 1):
   BIT  59    : EOF BIT
   BITS 56-58 : SPECIFICATION
   BIT  55    : CHAR/BINARY (CHAR = 0)
   BIT  54    : READ/WRITE BIT (READ = 0)
   BITS 48-53 : MASK FOR EOF-TEST
   BITS 18-47 : UNUSED
   BITS  0-17 : LRL (PASCAL RECORD LENGTH)

  INPUT (USED BY THE COMPILER AND THE USER):
          BSS     10              CHARACTER BUFFER
          VFD     60/*-10         POINTER USED BY IOPAK
          VFD     1/0,3/0,1/0,1/0,6/20B,48/1
                                  NOT EOF,IN,CHAR,READ,MASK FOR EOR,LRL=1
INFET     DATA    5LINPUT         NAME MAY BE CHANGED LATER
          VFD     60/INBUF,60/INBUF,60/INBUF
          VFD     60/INBUF+BUFLGTH

  OUTPUT (USED BY THE COMPILER ,THE USER, AND FQCOUT):
          BSS     10              CHARACTER BUFFER
          VFD     60/*-10         POINTER USED BY IOPAK
          VFD     1/1,3/2,1/0,1/1,6/30B,48/1
                                  EOF,OUT,CHAR,WRITE,MASK FOR EOF,LRL=1
OUTFET    DATA    6LOUTPUT        NAME MAY BE CHANGED LATER
          VFD     60/OUTBUF,60/OUTBUF,60/OUTBUF
          VFD     60/OUTBUF+BUFLGTH

  PASCAL (USED BY POSYS ONLY):
PASCAL    DATA    6LPASCAL
          VFD     60/HEADER,60/HEADER,60/HEADER
          VFD     60/HEADER+65    CORRECT LIMIT KNOWN AT RUN-TIME

  PASCLGO (USED BY POSYS ONLY):
PASCLGO   DATA    7LPASCLGO
          VFD     60/BIGBUF,60/BIGBUF,60/BIGBUF
          VFD     60/HEADER+65    CORRECT LIMIT KNOWN AT RUN-TIME

  THE FOLLOWING FET IS USED EITHER FOR THE KEEP FILE (IF ANY),
```

```
  OR FOR THE LOAD FILE (IF ANY), OR NOT AT ALL.
KEEPFET  DATA     0                NAME PASSED AS PARAMETER
         VFD      60/HEADER,60/HEADER,60/HEADER
         VFD      60/HEADER+65     CORRECT LIMIT KNOWN AT RUN-TIME.

LOADFILE EQU      KEEPFET

  PMDFILE (USED BY POSYS ONLY):
PMDFET   DATA     7LPMDFILE
         VFD      60/INBUF,60/INBUF,60/INBUF
         VFD      60/INBUF+BUFLGTH
         EJECT
  DECLARATIONS
  ------------------------------------------------------------------

  THE ADDRESS OF THE COMPILER PARAMETER RECORD:
PARMLIST EQU      BIGBUF+1

  DESCRIPTION (CF. HEADER):

ICO      EQU      PARMLIST   :   FIRST ADDRESS AVAILABLE TO USER PROGRAM
ERRFLAG  EQU      PARMLIST+1 :   ERROR FLAG
ERRBITS  EQU      PARMLIST+2 :   FOUR WORDS OF ERROR BITS
                  PARMLIST+6 :   LOAD POINT OF BINARY INFORMATION
                  PARMLIST+7 :   LIMIT OF CODE
                  PARMLIST+8 :   LOAD POINT OF JUMP TABLE
                  PARMLIST+9 :   EXTERNAL ROUTINES FLAGS
                  PARMLIST+10:   ENTRY POINT


ERRLIM   EQU      PARMLIST+6

  CERTAIN FLAGS:
FQCFLAG  DATA     0                ≠ 0 IFF FQC IS ACTIVE
FQCFILFL DATA     0                ≠ 0 IFF FQCFILE EXISTS
EXECFLAG DATA     0                = 1 IF A LOAD PROGRAM IS BEING EXECUTED,
                                   = 2 DURING EXECUTION OF A COMPILED PROGR.,
                                   = 0 OTHERWISE.

  FIELD LENGTH STATUS WORD:
F.LGTH   DATA     0

  CONSTANTS USED IN CONNECTION WITH THE PASCAL CALL CARD OPTIONS:
IONOGO   DATA     5LINPUT,6LOUTPUT,4LNOGO
FQCNAME  DATA     7LFQCFILE

  PP REQUEST WORDS:
ABTREQ   VFD      18/3LABT,6/0,36/6RNODUMP
RPVREQ   VFD      18/3LRPV,2/1,16/27B,6/0,18/RPVTAB

  CONCERNING CURRENT POSITION ON FILE PASCAL:
RECPOS   DATA     1                (POSYS HAVING NUMBER 0)
FQCPOS   EQU      8                POSITION OF FQCOUT ON PASCAL
ERRECPOS EQU      10               POSITION OF ERROR MESSAGES RECORD
PMDPOS   EQU      11               POSITION OF PMD ROUTINE

  BASE ADDRESS OF RUN-TIME STACK:
RTSBASE  DATA     0

  CERTAIN LENGTHS:
PMDLGTH  EQU      300B             LENGTH OF PMD ROUTINE
JMPTABLG EQU      50               LENGTH OF JUMP TABLE
CTRMAX   EQU      1600             MAX. NUMBER OF COUNTERS FOR FQC

  PARAMETER TABLE:
PARAMS   DATA     L=P=             PROGRAM FILE
```

```
PPAR      DATA    0
          DATA    L=L=             LISTING FILE
LPAR      DATA    0
          DATA    L=D=             DATA FILE
DPAR      DATA    0
          DATA    L=R=             RESULT FILE
RPAR      DATA    0
          DATA    L=OPT=           OPTION (NOGO OR FC OR FC*)
OPTPAR    DATA    0
          DATA    L=LOAD=          LOAD FILE
LOADPAR   DATA    0
          DATA    L=FL=            EXECUTION FL
FLPAR     DATA    0
          DATA    L=LL=            EXECUTION LINE LIMIT
LLPAR     DATA    0
          DATA    L=KEEPF=         KEEP FILE
KEEPFPAR  DATA    0
          DATA    L=FCL=           FREQUENCY COUNT LOWER
FCLPAR    DATA    0
          DATA    L=FCU=           FREQUENCY COUNT UPPER
FCUPAR    DATA    0
          DATA    0

  DAYFILE MESSAGES:
DATEMSG1  DIS     ,=- PASCALSYSTEM VERS. .........=
DATEMSG2  DIS     ,=- PGM. OF .........=
ENDCOMP   DIS     ,=- END COMPILATION=
ENDMSG    DIS     ,=- END PASCAL=
KEEPMSG   DIS     ,=- BIN. FILE KEPT ON .......=
ERRMSG1   DIS     ,=* ERROR IN PASCAL CARD=
ERRMSG2   DIS     ,=* KEYWORD USED TWICE=
ERRMSG3   DIS     ,=* ILLEGAL FL=
ERRMSG4   DIS     ,=* ILLEGAL LL=
ERRMSG5   DIS     ,=* COMPILER OR PROGRAM NEEDS CM......=
ERRMSG6   DIS     ,=* ERROR(S) IN PASCAL PROGRAM=
ERRMSG7   DIS     ,=* FL TOO SMALL=
ERRMSG8   DIS     ,=* BAD FCL OR FCU=
LOADMSG   DIS     ,=* LOADFILE BAD=
READMSG1  DIS     ,=* INTEGER INPUT EXPECTED=
READMSG2  DIS     ,=* BAD INPUT=
ADDRMSG   DIS     ,=* ADDR. = ......=
MEN       DIS     ,=* ERRORS IN I/O=
MER       DIS     ,=* ERROR IN RESET=
MEOUT     DIS     ,=* TRIED TO WRITE ON IN-FILE=
MEOF      DIS     ,=* TRIED TO READ PAST EOF=
LINEMSG   DIS     ,=* LINE LIMIT EXCEEDED=
ASSMSG    DIS     ,=* VALUE OUT OF RANGE=
INDXMSG   DIS     ,=* INDEX OR CASE VAR OUT OF RANGE=
DIVMSG    DIS     ,=* DIVISION BY ZERO=
OVFLMSG   DIS     ,=* RUN-TIME STACK OVERFLOW=
NOPMDMSG  DIS     ,=* PROGRAM TOO LARGE FOR PMD=
IMPPMD    DIS     ,=* PMD IMPOSSIBLE=
MODEMSG   DIS     ,=* MODE = 0=

CALL      MACRO   P+1
          EQ      P
          ENDM

WAIT      MACRO   I
  USES B1,I.      B1
          NZ      X.I,*
          ENDM

  PP REQUEST ROUTINE:
PPREQ     DATA    0
  ENTRY COND:
```

```
   X0 = PP ROUTINE NAME, A1/X1 = LFN + C/S,  X6 = FUNCTION CODE
   EXIT COND : A1 = FET-ADDR (UNCHANGED)
  USES A4,A0, X0,X4,X6.
         MX4       42                MASK FOR LFN
         BX4       X1*X4
         BX6       X6+X4
         SA6       A1                SET CODE IN FET
         LX0       42
         SX6       A1
         BX6       X6+X0
         MX0       1
         LX0       41                RECALL BIT
         BX6       X6+X0
         SA6       1
+        SA4       1
         NZ        X4,*
         EQ        PPREQ

  MESS (ISSUES MESSAGE REQUEST WITHOUT AUTO-RECALL).
  UPON ENTRY, X0 = POINTER TO MESSAGE.
  USES A2,A6, X0,X2,X6.
MESS     BSS       1
         SX6       3LMSG
         LX6       42
         BX6       X6+X0             COMBINE WITH MSG-ADDRESS
         SA6       1
         WAIT      2
         EQ        MESS

  MEMORY REQUEST ROUTINE.
  ENTRY COND: X1 = DESIRED FL, A2/X2 = F.LGTH.
  EXIT COND: X1 = X6 = [F.LGTH] = NEW FIELD LENGTH.
  USES X2, A3/X3, B1 = 1.
MEMREQ   BSS       1
         BX6       X1
         LX6       30
         SA6       A2

         MX2       1
         SX6       3RMEM
         LX6       42
         LX2       41
         BX6       X6+X2
         SX2       A2
         BX6       X6+X2
         SA6       B1
         WAIT      3
         BX6       X1
         SA6       A2
         EQ        MEMREQ

  SKIP FORWARD PASCAL ROUTINE:
  UPON ENTRY, X3 = SKIP COUNT.
  USES X0, A1/X1, A6/X6, B1 = 1.
SKIPF    BSS       1
         SA1       PASCAL
         MX0       42
         BX6       X1*X0
         SX0       242B              SKIPF
         BX6       X6+X0
         SA6       A1

         SX6       3RCIO
         LX6       42
         MX0       1
         LX0       41
```

```
          BX6     X6+X0           RECALL BIT
          LX3     13
          BX6     X6+X3           NUMBER OF RECORDS
          SX0     PASCAL
          BX6     X6+X0
          SA6     B1
          WAIT    1
          EQ      SKIPF

  PROCEDURE TRANSFORM BINARY NUMBER INTO ITS DISPLAY CODE FORM:
  CONVERTS B2 ≤ 10 OCTAL DIGITS TAKEN FROM THE LOWER END OF X5 INTO
  DISPLAY CODE AND STORES THE RESULT LEFTMOST INTO X6. X5 IS THEREBY
  SHIFTED LEFT BY (60 - 3*B2). B2 IS 0 UPON EXIT.
  USES X1,X2, B1 = 1, B3.
BIN.DISP BSS      1
          SX6     B0
          MX1     57              (COMPLEMENT OF) MASK
          SB3     33B             =0=
BD1       BX2     -X1*X5          EXTRACT OCTAL DIGIT
          SX2     X2+B3
          BX6     X6+X2
          LX6     54
          LX5     57              NEXT OCTAL DIGIT RIGHTMOST
          SB2     B2-B1
          NZ      B2,BD1
          EQ      BIN.DISP

  PROCEDURE TRANSFORM DISPLAY CODE NUMBER TO BINARY.
  UPON ENTRY, X1 = LEFT ADJUSTED, ZERO-FILLED DISPLAY CODE NUMBER.
  UPON EXIT, X6 = CONVERTED X1 IF NO ERROR ENCOUNTERED,
                  400...00B OTHERWISE.
  USES X0,X2,X3, B2.
D.BERR    MX6     1
DISP.BIN  BSS     1
          MX0     54              MASK
          SX6     B0
          SB2     -1R0
          EQ      D.B2
D.B1      SX3     X2-1R8          RESULT SHOULD BE < 0
          SX2     X2+B2           RESULT SHOULD BE ≥ 0
          BX3     -X2*X3
          PL      X3,D.BERR       ERROR IF X2 < 0 OR X3 ≥ 0
          LX6     3
          BX6     X6+X2
D.B2      LX1     6
          BX2     -X0*X1
          NZ      X2,D.B1
          EQ      DISP.BIN

  IF FQCFILE EXISTS, GIVE IT DISP CODE OUTPUT AND CLOSE/UNLOAD IT:
PRTFQC    BSS     1
          SA1     FQCFILFL
          ZR      X1,PRTFQC
          SA5     OUTFET
          SX0     40B             DISP CODE OUTPUT
          SA2     A5+B1
          LX0     24
          BX6     X2+X0
          SX0     B1
          LX0     18
          BX6     X6+X0           L = 1 IN [FET+1]
            SA6     A5+B1
          SA2     FQCNAME
          BX6     X2
          SA6     A5
          SX0     3RCLO
```

```
          SA1       A5
          SX6       172B            CLOSE/UNLOAD
          RJ        PPREQ
          BX6       X5
          SA6       A5
          EQ       PRTFQC
          EJECT
    PROCEDURE LOAD EXTERNAL ROUTINES:
    UPON ENTRY, A1/X1 = EXTVECT OR PMD,
                X2 = [EXTFLAGS] OR 1 (1 FOR PMD),
                X4 = LOAD POINT OF (FIRST) ROUTINE,
    UPON EXIT,  X6 = LIMIT OF LAST LOADED ROUTINE,
                [RTSBASE] = X6.
    USES ALL REGISTERS EXCEPT A3,A5. EXPECTS B1 = 1.

SAVEX2    BSS       1
SAVEX1    BSS       1
SAVEX4    BSS       1
NEXTX4    BSS       1

LOADEXTS  BSS       1

    INSPECT THE EXTERNALS FLAGS AND LOAD THE DESIRED ROUTINES:
          SB5       59              SHIFT COUNT
          SX6       OUTBUF
          SA6       PASCAL+2        IN
          SA6       A6+B1           OUT
          SX3       B0              COUNTER FOR SKIPF
LDEXTS1   LX0       B5,X2           CURRENTLY RELEVANT BIT LEFTMOST
          NG        X0,LOAD.ONE
          SX3       X3+B1
LDEXTS2   AX2       1
          SA1       X1
          NZ        X2,LDEXTS1

          BX6       X4
          SA6      RTSBASE
          EQ        LOADEXTS

    LOAD THE DESIRED EXTERNAL ROUTINE:
LOAD.ONE  BX6       X2
          SA6       SAVEX2
          BX6       X1
          SA6       SAVEX1
          BX6       X4
          SA6       SAVEX4

    UPDATE RECPOS AND SKIP FORWARD BY [X3]:
          SA1       RECPOS
          IX6       X1+X3
          SX6       X6+B1
          SA6       A1
          ZR        X3,LD.1
          RJ        SKIPF

    READ IN ONE RECORD FROM LIBRARY INTO OUTBUF:
LD.1      SX0       3RCIO
          SX7       OUTBUF
          SA7       PASCAL+2        IN
          SX6       12B             READ
          SA1       PASCAL
          RJ        PPREQ

    PREFIX TABLE
          SA4       SAVEX4          LOAD POINT OF ROUTINE AND, AS B4,
          SB4       X4              RELOCATION CONSTANT.
```

```
          SA1       OUTBUF          X1 := IDENTIF WORD
          LX1       24              WC RIGHTMOST

    PIDL TABLE
          SB2       X1+B1           WC+1
          SA1       A1+B2           X1 := IDENTIFICATION WORD
          SA2       A1+B1           X2 := NAME/LENGTH OF ROUTINE
          SX6       X2+B4           X6 := LIMIT OF ROUTINE
          SA2       F.LGTH
          IX2       X6-X2           RESULT MUST BE < 0
          PL        X2,FL.ERR
          SA6       NEXTX4          LOAD POINT OF NEXT ROUTINE

    ENTR TABLE:
ENTR      LX1       24              WC RIGHTMOST
          SB2       X1+B1           WC+1
          SA2       A1+B2
          LX2       23
          SB2       X2              B2 =  WC/2 = NUMBER OF ENTRY POINTS
          SB2       B0-B2           B2 IS USED AS COUNTER AND AS DISPLA-
                                    CEMENT RELATIVE TO THE CONTENTS OF X1
          SA1       SAVEX1          X1 = 1 + (ADDRESS OF LAST TRANSFER
                                    WORD FOR CURRENT ROUTINE)
ENTR.1    SA2       A2+2
          SX6       X2              X6 := LOAD ADDRESS OF ENTRY POINT
          IX6       X6+X4           RELOCATE
          SX2       048             EQ
          LX2       24
          BX6       X6+X2
          LX6       30
          SA6       X1+B2           TRANSFER WORD NOW CONTAINS THE COR-
                                    RECT JUMP INSTRUCTION
          SB2       B2+B1
          NE        B2,ENTR.1

    PROCESS TEXT TABLES:
          MX0       3
          LX0       4               TO MASK OUT RELOCATION BITS
          SA1       A2
          SA2       PASCAL+2        IN
          SB3       X2              LIMIT OF TEXT
          SB6       15
          LX5       B6,X4           FOR MIDDLE ADDRESS RELOCATION
          LX6       B6,X5           FOR UPPER ADDRESS RELOCATION

TEXTTABL  SB2       A1+B1
          EQ        B2,B3,ENDTEXT   RETURN IF END OF DECK REACHED
          SA1       B2              X1 := IDENTIFICATION WORD
          SA0       X1+B4           A0 := RELOCATED LOAD ADDRESS OF TEXT
          LX1       24
          SB7       X1-2            B7 := WC-2 (LIMIT FOR B6)
          SA1       A1+B1
          BX2       X1              X2 := RELOCATION WORD
          SB6       B0-B1           B6 IS USED SIMULTANEOUSLY AS COUNTER
                                    AND AS DISPLACEMENT RELATIVE TO A0.
TEXTWORD  SA1       A1+B1           X1 := TEXT WORD
          LX2       4               CURRENT RELOCATION BYTE RIGHTMOST
          BX3       X2*X0           X3 := RELOCATION BYTE
          SB6       B6+B1
          NZ        X3,RELOC        GOTO RELOC IF RELOCATION WANTED
          BX7       X1
          SA7       A0+B6           TEXT WORD LOADED
          LT        B6,B7,TEXTWORD
          EQ        TEXTTABL

RELOC     SB2       X3
```

```
          JP        *-1+B2              SWITCH

OOIO:     IX7       X1+X4               LOWER ADDRESS RELOCATED
          SA7       A0+B6               TEXT WORD LOADED
          LT        B6,B7,TEXTWORD
          EQ        TEXTTABL
OIOO:     IX7       X1+X5               MIDDLE ADDRESS RELOCATED
          SA7       A0+B6               TEXTWORD LOADED
          LT        B6,B7,TEXTWORD
          EQ        TEXTTABL
OIIO:     BSS       2                   IMPOSSIBLE COMBINATION
IOOO:     IX7       X1+X6               UPPER ADDRESS RELOCATED
          SA7       A0+B6               TEXT WORD LOADED
          LT        B6,B7,TEXTWORD
          EQ        TEXTTABL
IOIO:     IX7       X1+X4               LOWER ADDRESS RELOCATED
          IX7       X7+X6               UPPER ADDRESS RELOCATED
          SA7       A0+B6
          LT        B6,B7,TEXTWORD
          EQ        TEXTTABL

ENDTEXT   SA4       NEXTX4              X4 := LOAD POINT OF NEXT ROUTINE
          SA2       SAVEX2
          SA1       SAVEX1
          SX3       B0
          EQ        LDEXTS2

  THE FIELD LENGTH IS TOO SMALL FOR LOADING:
FL.ERR    SX0       ERRMSG7
          RJ        MESS
          EQ        ABN.EX2
          EJECT
  PROCEDURE LOAD
  UPON ENTRY, A1/X1 = FET OF PASCAL OR OF LOADFILE OR OF PASCLGO.

LOAD      BSS       1

  READ THE CODE AND THE JUMP TABLE INTO THE BUFFER:
          SA2       LOADPT
          SX0       3RCIO
          BX6       X2
          SA6       A1+2                IN
          SA6       A6+B1               OUT
          SX6       12B                 READ BINARY
          RJ        PPREQ

  SET [RTSBASE]:
          SA2      LIMCODE
          BX6      X2
          SA6       RTSBASE

  MOVE THE JUMP TABLE, IF ANY
          SA2       LPJMPTAB
          ZR        X2,LOAD

          SA3       A2-B1               X3 := ADDRESS OF FIRST JUMP WORD
          SB2       X2+B2BLG-1
          BX6       X4
          SA6       X2+B2
          SB2       B2-B1
          GE        B2,B0,*-1

          EQ        LOAD
          EJECT
  PROCEDURE FREQUENCY COUNTS:
  THIS PROCEDURE CONSISTS OF TWO PARTS, CPFQC1 AND CPFQC2.
```

```
  CPFQC1 IS CALLED BEFORE EXECUTION OF THE USER PROGRAM AND PASSES
  PARAMETERS TO FQC. (FQC ITSELF IS CALLED AND RECALLED BY PRO-
  CEDURE GO.) CPFQC2 LOADS AND EXECUTES FQCOUT.

FQCDAT   VFD      58/0,1/0,1/0
         BSS      1
         VFD      60/BIGBUF+5

BLKLGTH  BSS      1

  GET FCL AND FCU:
CPFQC1   BSS      1
         SA2      LPJMPTAB
         SX6      X2+JMPTABLG      DEFAULT FCL: BEGINNING OF CODE
         SA1      FCLPAR           LOWER BOUND OF CODE TO SUPERVISE
         ZR       X1,CPFQC1.1
         RJ       DISP.BIN         CONVERT TO BINARY
         NG       X6,FQCERR        CHECK FOR ERROR
CPFQC1.1 SA6      A1               STORE FCL BINARY
         BX5      X6               X5 := FCL
         SA2      LIMCODE          DEFAULT FCU: LIMIT OF CODE
         BX6      X2
         SA1      FCUPAR           UPPER BOUND OF CODE TO SUPERVISE
         ZR       X1,CPFQC1.2
         RJ       DISP.BIN         CONVERT TO BINARY
         NG       X6,FQCERR        CHECK FOR ERROR
CPFQC1.2 IX0      X5-X6            -(LENGTH OF CODE TO BE MONITORED)
         PL       X0,FQCERR        CHECK FOR LENGTH > 0
         BX0      -X0              X0 := +LENGTH

  AT THIS POINT X5 = FCL, X6 = FCU, X0 = X6-X5.
  FIND THE BLOCK LENGTH AND ITS 2LOG, I.E. INTEGERS BLKLGTH AND N
  SUCH THAT BLKLGTH = 2↑N AND (LENGTH OF CODE)/BLKLGTH ≤ (MAX. NUM-
  BER OF COUNTERS).
         SX1      CTRMAX/2
         SB1      1
         SB4      B4+B1
         IX2      X1-X0
         NG       X2,*-1
         SX4      B4               NOW X4 = B4 = N
         SX7      B1
         LX7      B4,X7
         SA7      BLKLGTH          NOW X7 = BLKLGTH

  ROUND LENGTH OF CODE TO NEXT HIGHER INTEGER MULTIPLE OF BLKLGTH
  AND ADJUST FCU:
         IX0      X0+X7
         SX0      X0-1
         AX0      B4,X0
         LX0      B4,X0
         IX6      X5+X0
         SA6      FCUPAR

  AT THIS POINT X4 = 2LOG(BLKLGTH), X5 = FCL < X6 = FCU AND
  X7 = BLKLGTH. IF FC* IS SPECIFIED, WE WANT [LPJMPTAB] + JMPTABLG ≤ FCL
  < FCU < [LIMCODE] + [BLKLGTH], OTHERWISE 0 ≤ FCL < FCU ≤ FIELD
  LENGTH.
         MX0      54
         SA1      OPTPAR
         LX1      18
         BX0      -X0*X1
         SX0      X0-1R*
         NZ       X0,CPFQC1.3      IF OPT=FC* THEN
         SA1      LPJMPTAB         BEGIN
         SX1      X1+JMPTABLG         X1 := [LPJMPTAB] + JMPTABLG.,
         SA2      LIMCODE             X2 := [LIMCODE] + 7
```

```
          IX2       X2+X7               END ELSE
          EQ        CPFQC1.4            BEGIN X1 := 0.,
CPFQC1.3  SX1       0                      X2 := FIELD LENGTH
          SA2       F.LGTH              END

  CHECK X1 ≤ X5, X6 ≥ X2:
CPFQC1.4  IX0       X5-X1
          NG        X0,FQCERR
          IX0       X2-X6
          NG        X0,FQCERR

  SET UP FQC DATA TABLE:
          LX4       48
          LX5       24
          BX6       X4+X6
          BX6       X5+X6
          SA6       FQCDAT+1
          EQ        CPFQC1

FQCERR    SX0       ERRMSG8
          RJ        MESS
          EQ        ABN.EX3

  ----------------------
CPFQC2    BSS       1

  SKIP FORWARD TO FREQUENCY COUNTS OUTPUT ROUTINE FQCOUT:
          SA1       RECPOS
          SX3       X1-FQCPOS
          SX6       HEADER
          SA6       PASCAL+2            IN
          SA6       A6+B1               OUT
          ZR        X3,CPFQC2.1
          BX3       -X3
          RJ        SKIPF

  READ THE HEADER:
CPFQC2.1  SX0       3RCIO
          SX6       12B                 READ
          SA1       PASCAL
          RJ        PPREQ

  RAISE THE FIELD LENGTH IF NECESSARY. LOAD.
          SA1       REQRDFL
          SA2       F.LGTH
          IX0       X2-X1
          PL        X0,CPFQC2.2
          RJ        MEMREQ
          SA6       PASCAL+4            LIMIT

CPFQC2.2  SA1       PASCAL
          RJ        LOAD
          SX6       FQCPOS+2            FQCOUT CONSISTS OF 2 RECORDS
          SA6       RECPOS

  SET LINE LIMIT. PASS FCL, FCU, FCKIND AND BLOCK LENGTH.
  EXECUTE.
          SA1       INFET-1
          MX0       1
          BX6       -X0*X1
          SA6       A1
          SX6       -B1
          SA6       LINECTR
          SB2       B0
          SA1       FCLPAR
          SA2       FCUPAR
```

```
          BX6       X1
          SA6       5001B
          BX6       X2
          SA6       5002B
          SA1       BLKLGTH
          BX6       X1
          SA6       5003B
          MX0       54
          SA1       OPTPAR
          LX1       18
          BX6       -X0*X1          NOW X6 = Ξ*Ξ OR X6 = 0
          SA6       5004B
          SB3       B0
          RJ        GO
          EQ        CPFQC2
          EJECT
    PROCEDURE GO.
    EXECUTES THE LOADED PROGRAM.
    USES ALL REGISTERS, BUT LEAVES B1 = 1 UNCHANGED.
    UPON ENTRY, B2 ≠ 0 IF FQC IS TO BE CALLED,
                B3 = VALUE TO STORE AT EXECFLAG.

RCLFQC    VFD       18/3LRCL,1/0,1/1,40/FQCDAT
GO        BSS       1

    ASK FOR A NEW PAGE IF THE OUTPUT FILEΞS NAME IS ΞOUTPUTΞ
          SA1       OUTFET
          SA2       IONOGO+1        X2 := ΞOUTPUTΞ
          MX0       42
          BX0       X1*X0           EXTRACT FILE NAME
          BX0       X2-X0           COMPARE WITH ΞOUTPUTΞ
          NZ        X0,GO.1

          SA1       OUTFET-2
          SX6       1R1
          SA6       X1
          SB7       A1+B1
          CALL      PUTC
          SA1       OUTFET-2
          SX6       0                 EOL
          SA6       X1
          SB7       A1+B1
          CALL      PUTC

    SET B4, B5, X7:
GO.1      SA1       F.LGTH
          SB4       X1-32           B4 := FIELD LENGTH - 32
          SX7       GO.3            RETURN ADDRESS
          SA1       LPJMPTAB
          SB5       X1+PMDLGTH+JMPTABLG
          SA1       RTSBASE
          SB7       X1
          GE        B5,B7,*+1
          SB5       B7
          SET B5 = MAX(CODELENGTH,PMDLGTH) + LIMIT OF MP DATA
          SA6       RTSBASE
          SA1       ENTRYPT
          SB7       X1

    IF B2 ≠ 0 THEN SET FQCFLAG AND CALL FQC:
          ZR        B2,GO.2
          SX6       1
          SA6       FQCFLAG
          SX6       3RFQC
          LX6       42
          MX0       1
```

```
          LX0       41
          BX6       X6+X0
          SX6       FQCDAT
          BX6       X6+X0
          SA6       B1
          WAIT      1
GO.2      SX6       B3
          SA6       EXECFLAG
          JP        B7

GO.3      SB1       1
          SX6       B0
          SA6       EXECFLAG
          SA1       FQCFLAG
          SB2       0

  IF [FQCFLAG] ≠ 0 THEN RECALL FQC, SET B2 := 1 AND
  SET [FQCFLAG] := 0 AGAIN:
          ZR        X1,GO
          SX7       2
          SA7       FQCDAT
          SA1       RCLFQC
          BX6       X1
          SA6       B1
          WAIT      1
          SX6       B0
          SA6       FQCFLAG
          SB2       1
          EQ        GO
          EJECT
  PROCEDURE KEEP
  PRODUCES A NEW LOADABLE BINARY DECK.
  UPON ENTRY, X1 CONTAINS THE NAME OF THE KEEP FILE, AND THE
  PROGRAM TO BE KEPT IS LOADED.
  USES    A1,A2,A3,A4,A6,A7, B1 = 1,B2,
       X0,X1,X2,X3,X4,X6,X7.

TIM.ANSW  BSS       1
LP.LPJMP  BSS       1

KEEP      BSS       1
          BX6       X1                NAME OF KEEP FILE
          SA6       KEEPFET
          SA6       KEEPMSG+2

  GET THE DATE:
          MX1       1
          LX1       25
          SX6       3RTIM
          LX6       42
          BX6       X6+X1
          SX1       TIM.ANSW
          BX6       X6+X1
          SA6       B1
          WAIT      2
          SA1       X1
          BX6       X1
          SA6       DATE

  SET [REQRDFL]:
          SA1       F.LGTH
          BX6       X1
          SA6       REQRDFL

  [LOADPT] := MIN([LOADPT],[LPJMPTAB]) ., [LPJMPTAB] := 0
          SA2       LPJMPTAB
```

```
          SA3       LOADPT
          IX0       X2-X3
          LX3       18
          BX6       X3+X2
          SA6       LP.LPJMP
          SX6       B0
          SA6       A2
          PL        X0,KEEP.1
          BX6       X2
          SA6       A3

    WRITE OUT THE HEADER:
KEEP.1    SX6       HEADER+7
          SA6       KEEPFET+2       IN
          SX6       HEADER
          SA6       A6+B1           OUT
          SA1       KEEPFET
          SX0       3RCIO
          SX6       26B             WRITER
          RJ        PPREQ

    WRITE OUT THE JUMP TABLE AND THE CODE:
          SA2       LOADPT
          SA3       A2+B1           LIMCODE
          BX6       X2
          SA6       KEEPFET+3       OUT
          BX6       X3
          SA6       A6-B1           IN
          SX0       3RCIO
          SX6       26B             WRITER
          SA1       KEEPFET
          RJ        PPREQ

    RESTORE [LOADPT] AND [LPJMPTAB] FOR EXECUTION:
          SA1       LP.LPJMP
          SX6       X1
          SA6       LPJMPTAB
          AX1       18
          BX6       X1
          SA6       LOADPT
    ISSUE A DAYFILE MESSAGE AND RETURN:
          SX0       KEEPMSG
          RJ        MESS
          EQ        KEEP
          EJECT
    PROCEDURE PRINT ERROR MESSAGES.
    WORDS [ERRBITS], [ERRBITS+1], ...,[ERRLIM-1] EACH CONTAIN 32 RIGHT-
    MOST ERROR BITS WHICH MAY BE SET BY THE COMPILER. THIS ROUTINE
    INSPECTS EACH SUCH BIT AND, ACCORDING TO WHETHER IT IS ON OR OFF,
    PRINTS OR SKIPS THE CORRESPONDING ERROR MESSAGE. THE ERROR MES-
    SAGES RESIDE ON RECORD NO. ERRECPOS ON PASCAL. THEY ALL HAVE THE FOL-
    LOWING FORMAT:
      WORD 1 :    LENGTH OF ACTUAL MESSAGE,
      WORDS 2..N: THE MESSAGE TEXT.
    THE TOTAL NUMBER OF MESSAGES (INCLUDING THE TITLE) IS = 0 (MOD 32).
    PRTERR ASSUMES THAT PUTB DOES NOT ALTER B5 OR B6.

SAVEB5B6  BSS       1
PRTERR    BSS       1

    SKIPF TO ERROR MESSAGE RECORD:
          SA3       RECPOS
          SX3       X3-ERRECPOS
          BX3       -X3
          RJ        SKIPF
```

```
  READ IN ERROR MESSAGES:
          SX6     ERRLIM
          SA6     PASCAL+2
          SA6     PASCAL+3
          SX6     3RCIO
          SX6     12B             READ
          SA1     PASCAL
          RJ      PPREQ

  SET ERROR BIT NUMBER 0 FOR TITLE:
          SA1     ERRBITS
          SX6     B1
          BX6     X6+X1

  PRINT ERROR MESSAGES:
          SA6     A1
          SB5     A1
          SB4     ERRLIM

XERRWORD  SA1     B5              X1 := NEXT ERROR BITS WORD
          LX1     27
          SB6     32
XBIT      LX0     B6,X1           CURRENTLY RELEVANT BIT LEFTMOST
          NG      X0,PRTMSG

  SKIP CURRENT ERROR MESSAGE, IF UNUSED:
          SA2     B4
          SB4     B4+X2
          SB4     B4+B1
          EQ      PRTERR1

  SAVE REGISTERS AND PRINT ERROR MESSAGE:
PRTMSG    BX6     X1
          SA6     B5
          SX6     B5
          LX6     18
          SX0     B6
          BX6     X6+X0
          SA6     SAVEB5B6

          SA1     B4              X1 := NUMBER OF MESSAGE WORDS
          SB6     X1
          SB5     B4+B6
          SB5     B5+B1           B5 := (ADDRESS OF LAST MESSAGE WORD) + 1

XMSGWORD  SA1     B5-B6           X1 := NEXT MESSAGE WORD
          BX6     X1
          SA1     OUTFET+2        IN
          SA6     X1
          SB7     OUTFET-1
          CALL    PUTB
          SB6     B6-B1
          NZ      B6,XMSGWORD

  RESTORE AND UPDATE REGISTERS:
          SB4     B5
          SA1     SAVEB5B6
          SB6     X1
          AX1     18
          SB5     X1
          SA1     B5

  ERROR BIT IS PROCESSED:
PRTERR1   SB6     B6-B1
          NZ      B6,XBIT
```

```
  ERROR BIT WORD IS PROCESSED:
        SB5     B5+B1
        SB7     B5-ERRLIM
        NZ      B7,XERRWORD

        SX6     OUTFET-12
        SA6     OUTFET-2        RESET POINTER
        EQ      PRTERR
        EJECT
  PROCEDURE LOAD AND EXECUTE PMD:
  UPON ENTRY, [SAVEX5] = ADDRESS WHERE ERROR OCCURRED,
              [SAVEB5] = POINTER TO BASE OF LATEST DATA SEGMENT.
  USES ALL REGISTERS. EXPECTS B1 = 1 AND THAT CHAR AND TEXT DO NOT
  ALTER B1 OR B4.

SAVEB5  BSS     1
SAVEX5  BSS     1

CALLPMD BSS     1

  RETURN IF B6 NOT WITHIN FIELD LENGTH:
        SA1     F.LGTH
        SB7     -B6
        SB7     X1+B7
        PL      B7,PMD.1
        SX0     IMPPMD
        RJ      MESS
        EQ      CALLPMD

  CHANGE THE OUTPUT FILE=S NAME TO =OUTPUT=:
PMD.1   SA1     IONOGO+1        X1 := =OUTPUT=
        BX6     X1
        SA6     OUTFET

  READ THE PMD FILE INTO THE INPUT BUFFER AND CHECK
  WHETHER END-OF-RECORD HAS BEEN REACHED:
        SX0     3ROPE
        SX6     142B            OPEN/REWIND FOR READ
        SA1     PMDFET
        RJ      PPREQ
        SX0     3RCIO
        SA1     PMDFET
        SX6     52B             REWIND
        RJ      PPREQ
        SX0     3RCIO
        SX6     12B             READ
        RJ      PPREQ
        SA1     A1              GET CODE/STATUS FOR CHECKING EOR BIT
        LX1     55              EOR BIT LEFTMOST
        NG      X1,PMD.3
        SX0     NOPMDMSG
        RJ      MESS
        EQ      CALLPMD

  SKIPF PASCAL TO PMD ROUTINE:
PMD.3   SX6     OUTBUF
        SA6     PASCAL+2        IN
        SA6     PASCAL+3        OUT
        SA1     RECPOS
        SX3     X1-PMDPOS
        SX6     PMDPOS+1
        SA6     A1
        BX3     -X3
        ZR      X3,*+1
        RJ      SKIPF
```

```
  CALL LOADEXTS FOR PMD. EXECUTE:
        SA4     LPJMPTAB        LOAD POINT OF PMD
        SX2     81
        SA1     PMD             ==TRANSFER WORD== FOR PMD
        RJ      LOADEXTS
        SA1     SAVEB5
        SA2     SAVEX5
        SB5     X1              B5 = BASE ADDRESS OF LATEST DATA SEGMENT
        SB6     X2              ADDRESS WHERE ERROR OCCURRED
        SX5     INBUF-1         ADDRESS OF FIRST ENTRY IN PMD TABLE - 1
        SA1     PMDFET+2        IN
        SX6     X1-1            ADDRESS OF LAST WORD IN PMD TABLE
        CALL    PMD
        SA1     OUTFET
        SX0     3RCIO
        SX6     26B             WRITER
        RJ      PPREQ

        EQ      CALLPMD
        EJECT
  IOPAK                                                 (BY R. SCHILD)
  ------------------------------------------------------------------

  FOR RETURN ADDRESSES:
RET2    BSS     1               USED BY GETB, PUTB
RET1    BSS     1               USED BY GETC, PUTC, EOR
RET0    BSS     1               USED BY OPEN,RESET,REWRITE,CLOSE,
                                 READ,TEXT,WRITE

 GETB MOVES POINTER TO NEXT BINARY INPUT RECORD, EXCEPT WHEN
  NOTHING HAS YET BEEN READ (IN = OUT), THEN POINTER IS NOT MOVED.
 IT IS CALLED BY GETC TO GET NEXT WORD OF 10 PACKED CHARS.
 IF EOF IS REACHED (OR EOR FOR =INPUT=), THE EOF BIT IN EFET IS SET,
  AND IF THE FILE SPEC IS EXT, THE WRITE BIT IS ALSO SET.
 REGS X0-X7 ARE USED
GETB    SA7     RET2
        SA1     B7
        NG      X1,EOFERR
        SX1     X1              LRL
        SA3     B7+3            IN
        SA2     B7+4            OUT
        BX0     X3-X2
        ZR      X0,GB1          JUMP IF IN = OUT
        IX2     X2+X1           STEP OUT-PTR
        SA4     B7+5            LIMIT
        SX4     X4
        BX4     X2-X4
        NZ      X4,GB1          JUMP IF OUT NOT= LIMIT
        SA2     B7+2            OUT := FIRST
GB1     SX7     X2    **** NOTE: GETC USES THIS VALUE IN X7 ****
        SA7     B7+4            OUT
        SA7     B7-1            P
GB2     IX0     X3-X7           IN - OUT
        NG      X0,GB3          RETURN IF IN < OUT
        IX0     X0-X1           (IN-OUT) - LRL
        NG      X0,GB4          JUMP IF IN-OUT < LRL
GB3     SA1     RET2
        SB7     X1
        JP      B7

GB4     SA1     B7+1            LFN, C/S
        SX0     1030B
        BX0     X0*X1
        SA4     B7
        LX4     6               X4 := 20B FOR INPUT
        AX4     54                    30B FOR ALL OTHER FILES
```

```
          IX0       X0-X4
          PL        X0,GB5          JUMP IF EOF OR EOI
          SX0       3LCIO
          SX6       12B             READ CODE
          RJ        PPREQ
          SA1       A1
          SX0       36000B
          BX0       X1*X0
          NZ        X0,ERRN         ERRORS
          SA1       B7
          SX1       X1              LRL
          SA3       A3              IN
          EQ        GB2

 SET EOF BIT.  IF EXTEND FILE, SET WRITE BIT.
GB5       SA4       B7
          AX4       56              CHECK SPEC (BIT 59 = 0)
          SX4       X4-4            EXT = 4
          SA1       B7              EOF,WRITE BITS
          MX0       1               EOF
          NZ        X4,GB6          JUMP IF NOT EXT
          SX0       418             EOF AND WRITE
          LX0       54              INTO CORRECT POSITION
GB6       BX6       X1+X0
          SA6       B7              SET BIT(S)
          EQ        GB3             RETURN
          EJECT
 PUTB MOVES POINTER TO NEXT BINARY OUTPUT RECORD.
 IT IS CALLED BY PUTC TO OUTPUT ONE WORD OF 10 PACKED CHARS.
  REGS X0-X7 ARE USED

PUTB      SA7       RET2
          SA1       B7
          SX1       X1              LRL
          SA2       B7+3            IN
          SA3       B7+4            OUT
          IX2       X2+X1           STEP IN-PTR
          SA4       B7+5
          SX4       X4              LIMIT
          BX4       X2-X4
          NZ        X4,PB1          JUMP IF IN NOT= LIMIT
          SA2       B7+2            IN := FIRST
PB1       SX7       X2
          SA7       B7+3            IN
          SA7       B7-1            P
          IX4       X3-X7           OUT - IN
          IX0       X1-X4           LRL - (OUT-IN)
          NG        X4,PB3          JUMP IF OUT < IN
          PL        X0,PB4          JUMP IF (OUT-IN) ≤ LRL
PB2       SA1       RET2
          SB7       X1
          JP        B7              RETURN

PB3       SA4       B7+2
          SX4       X4              FIRST
          IX0       X0+X4
          SA4       B7+5
          SX4       X4              LIMIT
          IX0       X0-X4           X0 = LRL - AVAILABLE SPACE
          NG        X0,PB2          RETURN IF ENOUGH SPACE
PB4       SA1       B7+1            LFN
          SX0       3LCIO
          SX6       16B             WRITE CODE
          RJ        PPREQ
          SA1       A1              GET C/S
          SX0       36000B
```

```
            BX0     X0*X1
            NZ      X0,ERRN
            EQ      PB2             RETURN
            EJECT
 GETC MOVES POINTER TO NEXT INPUT CHAR.
 BUFFER IS USED IN REVERSE DIRECTION. (FIRST IS AT B7-2)
 IF CHAR BUFFER IS USED UP, GETB IS CALLED
 USES X0-X7,A0

GETC        SA7     RET1
            SA1     B7-1            X1 := P-PTR
            SX2     B7-11           X2 := ADDR OF END OF BUFFER
            IX0     X2-X1
            ZR      X0,GC2          JUMP IF END REACHED
            SX6     X1-1            P := P - 1
            SX0     X0+1
GC1         SA3     X6              TEST NEXT CHAR
            NZ      X3,GC0          RETURN IF NOT EOL
            ZR      X0,GC2          IF END REACHED, GET NEXT WORD (0)
            SX6     X2              P := END OF BUFFER
GC0         SA6     B7-1            STORE POINTER TO CURRENT CHAR
            SA1     RET1
            SB7     X1
            JP      B7

GC00        SX6     B7-11           X6 := ADDR OF LAST CHAR
            EQ      GC0
 GET NEW WORD
GC2         SA0     B7
+           SX7     *+1
            EQ      GETB
            SB7     A0
 NOW X7 = PTR TO NEXT WORD
            SA1     B7
            NG      X1,GC00         RETURN IF EOF
            SA2     X7              X2 := WORD
            MX0     54
            BX6     -X0*X2          EXTRACT LAST CHAR
            SA6     B7-11           STORE IT
            DUP     9,3
            LX2     54
            BX6     -X0*X2
            SA6     A6+1
            SX6     A6              P := ADDR OF FIRST WORD
 NOTE X0 ≠ 0, SIGNIFYING: END NOT REACHED
            SX2     B7-11           X2 := ADDR OF END OF BUFFER
            EQ      GC1
            EJECT
 PUTC MOVES POINTER TO NEXT OUTPUT CHAR.  IF BUFFER IS FULL,
  IT IS PACKED AND PUTB IS CALLED TO OUTPUT IT.
  IF EOL HAS BEEN STORED, CHARS UPTO EOL ARE PACKED AND
  OUTPUT VIA PUTB.
 USES X0-X7,A0
LINECTR     BSS     1
PUTC        SA7     RET1
            SA1     B7-1            X1 := P-PTR
            SX2     B7-2            X2 := ADDR OF END OF BUFFER
            IX0     X1-X2
            SA3     X1              TEST LAST CHAR
            ZR      X3,PC3          JUMP IF EOL
            ZR      X0,PC1          JUMP IF END REACHED
            SX6     X1+1            P := P + 1
PC0         SA6     B7-1            STORE POINTER TO CURRENT CHAR
            SA1     RET1
            SB7     X1
            JP      B7
```

```
 CHAR BUFFER FULL. PACK AND STORE IN OUTPUT BUFFER
PC1     SA1     B7+3            X1 := IN-PTR
        SA3     B7-11           FIRST BUFFER CHAR
        SA2     A3+1            SECOND BUFFER CHAR
        LX3     6
        BX6     X2+X3
        DUP     8,3
        SA2     A2+1
        LX6     6
        BX6     X6+X2
        SA6     X1
        SA0     B7
+       SX7     *+1
        EQ      PUTB            NEXT WORD
        SB7     A0
        SX6     B7-11           P := ADDRESS OF FIRST WORD
        EQ      PC0
 EOL.  PACK CHAR BUFFER AND STORE IN OUTPUT BUFFER
 IF NECESSARY (N = 10), ADD ONE WORD OF ZEROES
PC3     SA0     B1              SAVE
        SB1     X0+9            COUNTER := N - 1
        SX6     B0              INITIALIZE PACKED WORD
        ZR      B1,PC5          JUMP IF EOL ONLY CHAR
        SA2     X1              A2 := P
PC4     SA2     A2-1            NEXT BUFFER CHAR
        BX6     X6+X2           PACK
        LX6     54              ROOM FOR NEXT CHAR
        SB1     B1-1            DECREASE COUNTER
        NZ      B1,PC4          LOOP IF NOT N-1 CHARS
 N - 1 CHARS HAVE NOW BEEN PACKED LEFTADJUSTED WITH ZERO FILL.
 NOW STORE WORD, TEST FOR TWO ZERO-BYTES AND STORE ADDITIONAL
 WORD IF NEEDED.
PC5     SB1     A0              RESTORE
        SA2     B7+3            X2 := IN-PTR
        SA6     X2              STORE WORD IN OUTPUT BUFFER
        SA0     B7
        MX0     48              MASK
        BX0     -X0*X6          TEST FOR TWO ZEROES
        ZR      X0,PC6          JUMP IF ADDTL WORD NOT NEEDED
+       SX7     *+1
        EQ      PUTB
        SB7     A0
        SA2     B7+3            X2 := IN-PTR
        SX6     B0              ZEROES
        SA6     X2              STORE ZEROES IN OUTPUT BUFFER
PC6     SX7     *+1
        EQ      PUTB
        SB7     A0
        SA1     LINECTR
        ZR      X1,LINERR
        SX6     X1-1
        SA6     A1
        SX6     B7-11           P := ADDR OF FIRST WORD
        EQ      PC0             RETURN
        EJECT
 EOR IS CALLED ONLY IF WRITE BIT SET, BY RESET, CLOSE AND POSYS.2,3,4 .
 CHECKS CHAR BUFFER AND EMPTIES IT IF NECESSARY.
 WRITES EOR.
  USES A0,X0-X4,X6,X7
EOR     SA7     RET1
        SA1     B7              C/B
        LX1     4               CHECK CHAR/BIN
        NG      X1,E2           JUMP IF NOT CHAR FILE
        SA2     B7-1            X2 := P-PTR
        SX3     B7-11           X3 := ADDR OF START OF BUFFER
        IX0     X2-X3
```

```
          ZR        X0,E2           JUMP IF EMPTY
          SX6       B0              INITIALIZE PACKED WORD
          SA4       X2              A4 := P
E1.1      SA4       A4-1            NEXT BUFFER CHAR
          BX6       X6+X4           PACK
          LX6       54              ROOM FOR NEXT CHAR
          SX0       X0-1            DECREASE COUNTER
          NZ        X0,E1.1         JUMP IF NOT N CHARS
          SA4       B7+3            X4 := IN
          SA6       X4              STORE WORD IN OUTPUT BUFFER
          SA0       B7
          MX0       48              MASK
          BX0       -X0*X6          TEST FOR TWO ZEROES
          SX7       *+1             JUMP IF ADDTL. WORD NOT NEEDED
          EQ        PUTB
          SB7       A0
          SA4       B7+3
          SX6       B0
          SA6       X4              STORE ZEROES IN OUTPUT BUFFER
E1.2      SX7       *+1
          EQ        PUTB
          SB7       A0
E2        SA1       B7+1
          SX6       26B             WRITE EOR
          SX0       3LCIO
          RJ        PPREQ
          SA1       A1
          SX0       36000B
          BX0       X0*X1
          NZ        X0,ERREOR       ERRORS
          SA1       RET1
          SB7       X1
          JP        B7
          EJECT
 OPEN CALLS OPE-ROUTINE
  SETS DISP CODE FOR PRINT AND PUNCH FILES
  INITIALIZES IN, OUT, AND P (AT B7-1)
  USES A0,X0,X1,X2,X4,X6,X7
OPEN      SA7       RET0
          SA1       B7+2            FIRST
          SX6       X1
          SA6       A1              CLEAR DEVICE TYPE AND DISP. CODE
          SA6       B7+3            IN
          SA6       B7+4            OUT
          SA1       B7
          LX1       4
          MX0       57
          BX0       -X0*X1          GET FILE SPEC BITS
          SX7       418             EOF AND WRITE BITS
          LX7       54              INTO CORRECT POSITION
          SA1       B7              EFET
          SA0       B2              SAVE
          SB2       X0+1
          JP        *+B2            SPEC = 7 NOT ALLOWED
          EQ        OINP
          EQ        OPRT
          EQ        OOUT
          EQ        OPCH
          EQ        OEXT
   SPEC = 6 NOT USED
OSCR      SX6       106B            OPEN WRITE, NO REWIND
          BX7       X7+X1           SET EOF AND WRITE BITS
OP        SA7       B7              STORE EFET WORD
          SX0       3LOPE
          SA1       B7+1
          RJ        PPREQ
```

```
            SB2     A0
            CALL    SETPTRS
            SA1     RET0
            SB7     X1
            JP      B7

    OOUT    EQU     OSCR
    OEXT    SX6     106B            OPEN WRITE, NO REWIND
            BX7     -X7*X1          CLEAR EOF AND WRITE BITS
            EQ      OP
    OINP    SX6     102B            OPEN READ,  NO REWIND
            BX7     -X7*X1          CLEAR EOF AD WRITE BITS
            EQ      OP
    OPCH    SA1     B7+2
            MX0     1
            LX0     19              FET-SIZE = 6
            BX6     X1+X0
            LX0     9               DISP = 10B (PUNCH)
            BX6     X6+X0
            SA6     A1
            SA1     B7
            EQ      OSCR
    OPRT    SA1     B7+2
            MX0     1
            LX0     19              FET-SIZE = 6
            BX6     X1+X0
            LX0     11              DISP = 40B (PRINT)
            BX6     X6+X0
            SA6     A1
            SA1     B7
            EQ      OSCR
            EJECT
     SETPTRS IS CALLED BY OPEN,RESET,REWRITE, AND POSYS.3
    SETPTRS SA7     RET1
            SA1     B7+2            FIRST
            SX6     X1
            SA6     B7+3            IN := FIRST
            SA6     B7+4            OUT := FIRST
            SA6     B7-1            P := IN (FOR BINARY FILES)
            SA1     B7              C/B
            LX1     4               TEST IF CHAR FILE
            NG      X1,SETP1        RETURN IF NOT CHAR
            SX6     B7-11
            SA6     B7-1            P := ADDR OF FIRST CHAR
    SETP1   SA1     RET1
            SB7     X1
            JP      B7
            EJECT
     RESET (FOR INPUT) CALLS EOR IF WRITE BIT SET.
     RESETS FILE,CLEARS EOF AND WRITE BITS, SETS POINTERS
     BY CALLING SETPTRS.
      USES A0,X0-X4,X6,X7
    SAVEB7  BSS     1               USED ALSO BY CLOSE
    RESET   SA7     RET0
            SA1     B7
            LX1     5               TEST WRITE BIT
            PL      X1,R1           JUMP IF READ
            SX6     B7
            SA6     SAVEB7
            CALL    EOR
            SA1     SAVEB7
            SB7     X1
    R1      SA1     B7
            SX0     41B
            LX0     54
            BX6     -X0*X1          CLEAR EOF AND WRITE BITS
```

```
          SA6       B7
 COMMON END FOR RESET AND REWRITE
RE        SX6       3LCIO
          SA1       B7+1
          SX6       52B             REWIND CODE
          RJ        PPREQ
          SA1       A1
          SX0       36000B
          BX0       X1*X0
          NZ        X0,ERRR         ERRORS
          SX0       1000B
          BX6       -X0*X1
          SA6       A1              CLEAR BEGINNING-OF-INFO-BIT
          CALL      SETPTRS
          SA1       RET0
          SB7       X1
          JP        B7
          EJECT
 REWRITE (RESET FOR OUTPUT) CHECKS FOR NON-IN-FILE.
 RESETS FILE, SETS EOF AND WRITE BITS, SETS POINTERS
 BY CALLING SETPTRS.
  USES A0,X0-X4,X6,X7
REWRITE   SA7       RET0
          SX0       41B
          LX0       54
          SA1       B7
          BX6       X0+X1           EOF AND WRITE BITS
          LX1       1
          AX1       57              CHECK FOR IN-FILE
          ZR        X1,ERROUT       ATTEMPT TO WRITE ON IN-FILE
          SA6       B7              SET EOF AND WRITE BITS
          EQ        RE              RESET FILE AND SET POINTERS
          EJECT
 CLOSE CALLS EOR IF WRITE BIT SET.
  UNLOADS SCRATCH,PRINT,PUNCH FILES, CLOSES IN,OUT,EXT FILES
  USES X0,X1,X2,X4,X6,X7,A0
CLOSE     SA7       RET0
          SA1       B7
          LX1       5               TEST WRITE BIT
          PL        X1,CL1          JUMP IF READ
          SX6       B7
          SA6       SAVEB7
          CALL      EOR
          SA1       SAVEB7
          SB7       X1
CL1       SA1       B7
          SX6       132B            CLOSE, NO REWIND (IN,OUT,EXT)
          LX1       3               TEST LOW BIT OF SPEC
          PL        X1,CL2
          SX6       172B            CLOSE UNLOAD (PRINT,PUNCH,SCRATCH)
CL2       SA1       B7+1
          SX0       3LCLO
          RJ        PPREQ
          SA1       RET0
          SB7       X1
          JP        B7
          EJECT
 WEOR CAN BE CALLED BY THE USER PROGRAM TO WRITE AN END-OF-RECORD
WEOR      SA7       RET0
          SA1       B7+1
          SX0       3LCIO
          SX6       26B             WRITE-EOR CODE
          RJ        PPREQ
          SA1       A1
          SX0       36000B
          BX0       X0*X1
```

```
          NZ        X0,ERRN1        JUMP IF ERRORS
          SA1       RET0
          SB7       X1
          JP        B7
          EJECT
 READ  (READS PASCAL NUMBERS)                          (BY R.SCHILD)
 ----------------------------------------------------------------------

 ENTRY COND:
          X7 = RETURN ADDRESS
          X1 = ADDR OF VARIABLE
          B7 = TYPE
                    0 = INTEGER
                    1 = REAL

 EXIT COND :
          X6 = VALUE (STORED AT ADDRESS X1)

 IF B7 = 0 AN INTEGER MUST BE READ
 IF B7 = 1 REAL OR INTEGER MAY BE READ.
   AN INTEGER WILL BE CONVERTED

NEXTCH    MACRO
          SB7       INFET-1
+         SX7       *+1
          EQ        GETC
          SA1       INFET-2
          SA1       X1
          ENDM

LOADX     MACRO
          SA4       SVX6
          BX6       X4
          SA4       SVX4
          SA5       SVX5
          ENDM

STOREX    MACRO
          SA6       SVX6
          BX6       X4
          SA6       SVX4
          BX6       X5
          SA6       SVX5
          ENDM

VARADDR   BSS       1
VARTYPE   BSS       1

READ      SA7       RET0
          SA2       INFET-1
          NG        X2,EOFERR     JUMP IF ALREADY EOF
          BX7       X1
          SA7       VARADDR          SAVE ADDR OF VARIABLE
          SX7       B7
          SA7       VARTYPE          SAVE TYPE
          MX0       42
          SX7       B1
          BX7       -X0*X7
          LX7       18
          SX1       B2
          BX1       -X0*X1
          BX7       X7+X1
          SA7       SAVEB1B2
          SB2       -1               INDICATE SIGN NOT YET READ
          SA1       INFET-2
          SA1       X1               X1 := INPUT↑
```

```
            EQ      SK1
SKIP        NEXTCH
            SA2     INPUT-1
            NG      X2,STOR         JUMP IF EOF
SK1         ZR      X1,SKIP         EOL
            SB7     X1-55B
            ZR      B7,SKIP         BLANK
            SX1     X1-27
            NG      X1,ERRRD        LETTER
            GE      B2,B0,NOSIGN    JUMP IF SIGN ALREADY READ
            SB7     X1-10
            NG      B7,GETNUM       JUMP IF DIGIT
            SB7     X1-11
            GT      B7,ERRRD        SPEC CHAR, NOT SIGN
            SB2     -B7             0 = - , 1 = +
            EQ      SKIP
NOSIGN      SB7     X1-9
            GT      B7,ERRRD        SPEC CHAR
GETNUM      SX6     B0              X6 = Z := 0
            MX5     12              MASK / X5 < 0 : INTEGER
            SX4     B0              X4 = SCALE := 0
GN1         LX6     1
            IX1     X6+X1
            LX6     2
            IX6     X6+X1           Z := Z*10 + D
            BX1     X5*X6
            NZ      X1,ERROVFL      Z ≥ 2↑48
            STOREX
            NEXTCH
            LOADX
            SB7     X1-1RE
            ZR      B7,EXP10        E
            SX1     X1-1R0
            NG      X1,FIN          LETTER OR EOL
            SB7     X1-10
            NG      B7,GN1          DIGIT
            SB7     X1-20
            NZ      B7,FIN          NOT .
  DECIMAL POINT HAS BEEN READ
            PX6     B0,X6
            NX6     B0,X6           CHANGE INTEGER TO REAL
            SA5     C10             10.0 / X5 > 0 : REAL
GN2         STOREX
            NEXTCH
            LOADX
            SB7     X1-1RE
            ZR      B7,EXP10        E
            SX1     X1-1R0
            NG      X1,NOEX         LETTER OR DIGIT
            SB7     X1-9
            GT      B7,NOEX         NOT DIGIT
            FX6     X6*X5           Z := Z*10.0
            PX1     B0,X1
            FX6     X6+X1           Z := Z + D
            NX6     B0,X6
            SX4     X4+1            SCALE := SCALE + 1
            EQ      GN2
NOEX        ZR      X4,ERRRD        NUMBER ENDS WITH .
            MX3     60              X3 = SIGN(SCALE) := -
            EQ      GEX3
  EXPONENT PART
EXP10       PL      X5,EXP11        JUMP IF REAL
            PX6     B0,X6           CHANGE INTEGER TO REAL
            NX6     B0,X6
            SA5     C10             10.0 / X5 > 0 : REAL
            EQ      EXP12
```

```
EXP11    ZR       X4,ERRRD        NUMBER ENDS WITH .E
EXP12    SB1      -1              B1 = SIGN(EXP) := +
         STOREX
         NEXTCH
         SX1      X1-27
         NG       X1,ERRRD        LETTER OR EOL AFTER E
         SB7      X1-10
         NG       B7,GETEXP       JUMP IF DIGIT
         SB1      X1-11           SIGN(EXP), 0 = - , -1 = +
         GT       B1,ERRRD        SPEC CHAR AFTER E (NOT SIGN)
         NEXTCH
         SX1      X1-27
         NG       X1,ERRRD        LETTER OR EOL AFTER SIGN
         SB7      X1-9
         GT       B7,ERRRD        SPEC CHAR AFTER SIGN
GETEXP   BX3      X1              X3 = EXP := D
GEX1     BX7      X3
         SA7      SVX3
         NEXTCH
         SA3      SVX3
         SX1      X1-27
         NG       X1,FINEX        LETTER OR EOL
         SB7      X1-9
         GT       B7,FINEX        SPEC CHAR
         LX3      1
         IX1      X3+X1
         LX3      2
         IX3      X3+X1           EXP := EXP*10 + D
         EQ       GEX1
FINEX    LOADX
         NG       B1,GEX2         JUMP IF SIGN(EXP) = +
         IX4      X3+X4           SCALE := EXP + SCALE
         MX3      60              X3 = SIGN(SCALE) := -
         EQ       GEX3
GEX2     IX4      X3-X4           SCALE := EXP - SCALE
         BX3      X4
         AX3      59              X3 := SIGN(SCALE)
         BX4      X3-X4           SCALE := ABS(SCALE)
 AT GEX3:         X3 = SIGN(SCALE)    X4 = SCALE
                  X2 = SIGN(Z)  X6 = Z   X5 = FF = 10.0

  NOW COMPUTE 10↑SCALE
GEX3     SA1      C1.0            X1 = FAC := 1.0
         ZR       X4,GEX6
         SB7      59
GEX4     LX7      B7,X4           TEST ODD(SCALE)
         PL       X7,GEX5         JUMP IF EVEN
         FX1      X1*X5           FAC := FAC*FF
GEX5     FX5      X5*X5           FF := FF*FF
         AX4      1               SCALE := SCALE DIV 2
         NZ       X4,GEX4
GEX6     NG       X3,GEX7
         RX6      X6*X1           Z := Z*FAC
         EQ       FIN
GEX7     RX6      X6/X1           Z := Z/FAC

FIN      SA2      VARTYPE
         NZ       B2,FIN1         JUMP IF SIGN(Z) = +
         SX1      B0
         IX6      X1-X6           Z := -Z
FIN1     PL       X5,FIN2         JUMP IF REAL NUMBER READ
         ZR       X2,STOR         JUMP IF INTEGER EXPECTED
         PX6      X6
         NX6      X6              CONVERT TO REAL
STOR     SA1      VARADDR
         SA6      X1              STORE VALUE
```

```
            SA1     SAVEB182        RESTORE
            SB2     X1
            AX1     18
            SB1     X1
            SA1     RET0
            SB7     X1
            JP      B7

FIN2        NZ      X2,STOR         JUMP IF REAL EXPECTED
            EQ      ERRTYP          INTEGER EXPECTED,REAL READ
SVX3        BSS     1
SVX4        BSS     1
SVX5        BSS     1
SVX6        BSS     1
SAVEB182    BSS     1
            EJECT
TXT.ADR     BSS     1
TXT.CTR     BSS     1
TXT.CH      BSS     1
TXT.WD      BSS     1
TEXT        SA7     RET0
            X7 = RETURN ADDR.
            X1 = ADDR. OF FIRST TEXTWORD
            BX6     X1
            SA6     TXT.ADR
            SX6     B0
            SA6     TXT.CTR
TXT1        SA1     TXT.CTR
            BX6     X1
            NZ      X6,TXT2
            SA1     TXT.ADR         GET NEXT WORD
            SA2     X1
            BX6     X2
            SA6     TXT.WD
            SX6     A2+1
            SA6     TXT.ADR
            SX6     10
            SA6     TXT.CTR         RESET COUNTER
TXT2        SX6     X6-1
            SA6     TXT.CTR
            SA1     TXT.WD
            MX0     54
            LX1     6
            BX6     -X0*X1
            ZR      X6,TXT3         RETURN IF EOL
            SA2     OUTFET-2
            SA6     X2
            BX6     X1
            SA6     TXT.WD          STORE SHIFTED WORD
            SB7     OUTFET-1
            SX7     TXT1
            EQ      PUTC
TXT3        SA1     RET0
            SB7     X1
            JP      B7
            EJECT
  ERROR EXITS:

EOFERR      SX0     MEOF
            EQ      ERSV

ERRN        SX0     MEN
ERSV        SA1     RET2
            BX7     X1
            SX2     X1-BIGBUF
            PL      X2,ABN.EX1
```

```
            SA1     RET1
            BX7     X1
            SX2     X1-BIGBUF
            PL      X2,ABN.EX1
            SA1     RET0
            BX7     X1
            EQ      ABN.EX1

ERRN1       SA1     RET0
            SX7     X1
            SX0     MEN
            EQ      ABN.EX1

ERRR        SX0     MER
            SA1     RET0
            SX7     X1
            EQ      ABN.EX1

ERROUT      SX0     MEOUT
            SA1     RET0
            SX7     X1
            EQ      ABN.EX1

ERREOR      SX0     MEN
            SA1     RET1
            SX7     X1
            EQ      ABN.EX1

ERRTYP      SA1     RET0
            SX0     READMSG1
            BX7     X1
            EQ      ABN.EX1

ERRRD       SX0     READMSG2
            SA1     RET0
            BX7     X1
            EQ      ABN.EX1
ERROVFL     EQU     ERRRD

LINERR      SX6     -1
            SA6     LINECTR
            SX6     B7-11
            SA6     B7-1
            SX0     LINEMSG
            EQ      ERSV
            EJECT
  WRITE FORMATS PASCAL OUTPUT                    (BY K. JENSEN AND R. SCHILD)
  ----------------------------------------------------------------------------

WRITE       EQU     *

 ENTRY COND:
  X1 = VALUE TO BE PRINTED
  X2 = FIELDWIDTH
 (X3 = PLACES AFTER DECIMAL POINT, FOR FREAL ONLY)

OK          MACRO
            SA7     RET0
            NG      X2,BAD
            ZR      X2,WRTFIN           RETURN IF 0 FIELDWIDTH
            SA5     LMAX
            IX5     X2-X5
            PL      X5,BAD
            ENDM

WROUT       MACRO   R
```

```
            SA4     OUTFET-2
            S*R     X4
            SB7     OUTFET-1
            CALL    PUTC
            ENDM

FETCHOUT    MACRO   P
            SA4     OUTFET-2
            SA3     SIGN+P
            BX6     X3
            SA6     X4
            SB7     OUTFET-1
            CALL    PUTC
            ENDM

BLANKS      MACRO   K
            LOCAL   REPB
REPB        SX6     1R
            WROUT   A6                  *USES B7
            S*K     K-1
            NE      K,REPB
            ENDM

SSIGN        MACRO
            LOCAL   M1
            SA3     NEGA
            SB2     B2+1
            SX7     1R
            ZR      X3,M1
            SX7     1R-
M1          SA7     SIGN+B2
            ENDM

FINIS       MACRO
            SA4     RET0
            SB7     X4
            JP      B7
            ENDM

BAD         SA1      BARG
            SX2     10
            EQ      ALF

UNDEF       SA1     NDEF
            EQ      ALF

STAR        SB3     X2              * WROUT DESTROYS THE X REGISTERS
            SX6     1R
REP         WROUT   A6
            SX6     1R*
            SB3     B3-1
            GT      B3,REP
            FINIS

MOST        SX6     X6+33B
            SA6     SIGN+B2
RJEDI       BSS     1
            SB2     1               *B2:NO. OF DIGITS AND INDEX TO STORAGE
            SA5     C10             *10.0
SNEXT       PX3     B0,X4
            FX3     X3/X5
            UX3     B7,X3
            LX3     B7,X3            X3<-X4 DIV 10
            BX6     X4
            ZR      X3,MOST
            BX4     X3              *STORE NOW,FOR X3 IS DESTROYED BELOW
```

```
          LX3     1
          IX6     X6-X3
          LX3     2
          IX6     X6-X3             *X6<-X4-X3*10 (OLD X4)
          SX6     X6+33B
          SA6     SIGN+B2
          SB2     B2+1
          EQ      SNEXT

CONT      ZR      X4,PTZERO
          FX4     X4/X5
BISECT    BSS     1                 *IN: X1:VALUE. OUT: X0:SIGN OF S
                                    *                   X1:ABS(VALUE)
                                    *                   X2:UNALTERED,=FIELD L.
                                    *                   X3:USED,BUT NOW FREE
                                    *                   X4:ABS(.FRACTIONAL PART)
                                    *                   X5:USED,BUT NOW FREE
                                    *                   X6:S,X1=X4*10**S+1
                                    *                   X7:USED,BUT NOW FREE
          BX0     X1
          AX0     59
          BX1     X0-X1             *X1=ABS(X1)
          UX0     B7,X1
          SX0     B7
          PX0     B0,X0
          SA3     LOG2
          FX0     X0*X3
          UX6     B7,X0
          LX6     B7,X6
+         PL      X6,*+1
          SX6     X6-1
          SX6     X6+14
          BX0     X6
          AX0     59
          BX6     X0-X6
          BX3     X6
          SA4     C1.0
          ZR      X3,SCL2
          SA5     C10
          SB7     59
SCL       LX7     B7,X3
          PL      X7,SCL1
          FX4     X4*X5
SCL1      FX5     X5*X5
          AX3     1
          NZ      X3,SCL
SCL2      NG      X0,NEG
          SA5     C10
          FX4      X1/X4
          IX3     X4-X5
          NG      X3,CONT
          FX4     X4/X5
          SX6     X6+1
          EQ      CONT
NEG       FX4     X1*X4
          SA5     C10
          IX3     X4-X5
          NG      X3,CONT
          FX4      X4/X5
          SX6     X6-1
          NZ      X6,CONT
          SX0     B0
          EQ      CONT

FILL1     SX6     X3+1
          SA6     SIGN
```

```
          SX6     2
FILL      BSS     1
          SB3     B7                ↑SAVE Q
          SA3     P+B7              ↑X3:=10↑B7
          FX4     X4*X3             ↑X4:=X4*10↑B7
          SA3     HALF              ↑=0.5
          FX4     X4+X3             ↑ADD 1/2 TO ACCOMPLISH THE ROUND
          UX4     B7,X4
          LX4     B7,X4
          RJ      RUEDI             ↑STORE X4 IN PLACES SIGN+1 TO SIGN+B2
          SX6     1
          EQ      B2,B3,FILL        ↑ELSE WE,VE ROUNDED ALL 9,S
                                    ↑HENCE MUST INC EXPONENT
          SA3     SIGN-1
          SX6     X3-45B
          SA3     SIGN
          ZR      X6,FILL1
          SX3     X3-2
          PL      X3,FILL1
          SX6     1R+
          SA6     SIGN-1
          EQ      FILL1

PTZERO    SB3     X2-1
          ZR      B3,PTZ1
          BLANKS  B3
PTZ1      SX6     33B               ↑Ξ0Ξ
          WROUT   A6
          FINIS

NOE       SX6     1RE
          SA3     SIGN-1
          SX0     X3-45B
          ZR      X0,NOE1
          SX6     1R-
          EQ      NOE1
PTEXPO    SA1     SIGN              ↑FETCH EXPO
          SX3     100
          IX3     X1-X3
          PL      X3,NOE
          SX6      1RE              ↑ΞEΞ
          WROUT   A6
          SA3     SIGN-1
          BX6     X3
NOE1      WROUT   A6                ↑WRITE Ξ+Ξ OR Ξ-Ξ
          SA4     SIGN              ↑FETCH EXPO AGAIN
          SX3     X4-10             ↑JUMP IF EXPO ≥10
          PL      X3,ASINT          ↑TREAT AS INTEGER
          SX6     33B               ↑Ξ0Ξ
          WROUT   A6                ↑WRITE AS E+0^ OR E-0X
          SA3     SIGN              ↑FETCH EXPO AGAIN
          SX6     X3+33B            ↑ΞDIGITΞ
          WROUT   A6
          FINIS
ASINT     RJ      RUEDI
AGAIN     FETCHOUT B2
          SB2     B2-1
          NE      B2,AGAIN
          FINIS

CHAR      OK
          BX6     X1
CBEGIN    SB3     X2-1
          EQ      B3,CH1
          SA6     SIGN              ↑MUST STORE FOR BLANKS DESTROYS ALL X-REG.
          BLANKS  B3
```

```
          FETCHOUT B0
          FINIS
CH1       WROUT  A6                 ↱HAPPENS  WHEN X2=1
          FINIS

ALF       OK
          SB3    X2-10              ↱IF B3>0,B3=NO. OF BLKS.
          SB2    X2                 ↱SAVE
          LE     B3,POUT
          SB2    10                 ↱B2=NO OF CHARS TO BE OUTPUT
POUT      SB7    B2                 ↱B7=INDEX TO FILL
          MX0    54
OUT1      LX1    6
          BX6    -X0*X1
          SA6    SIGN+B7
          SB7    B7-1
          NE     B7,OUT1
          LE     B3,ALF1
          BLANKS B3
ALF1      FETCHOUT B2
          SB2    B2-1
          NE     B2,ALF1
          FINIS

BOOL      OK                        ↱IF X2<6,WILL WRITE ΞTΞ OR ΞFΞ
          SB3    X2-5               ↱IF >0,WILL WRITE THE WHOLE WORD
          GT     B3,BOOL1
          SX6    1RT                ↱ΞTΞ
          NZ     X1,CBEGIN          ↱IF X1=TRUE
          SX6    1RF                ↱ΞFΞ
          EQ     CBEGIN
BOOL1     SA4    TORF               ↱X4: Ξ TRUEFALSEΞ
          SB2    5                  ↱WANT THE FIRST FIVE CHAR.
          NZ     X1,BOOL2           ↱JUMP IF TRUE
          LX4    30                 ↱X4: ΞFALSE TRUEΞ
BOOL2     BX1    X4
          EQ     POUT               ↱WRITE FIRST B2 CHAR OF X1,PRECEDED BY B3 BLK

INTGR     OK
          ZR     X1,PTZERO
          SB3    X2
          BX4    X1
          AX4    48
          NZ     X4,UNDEF           ↱IS X1 TOO LARGE
          BX4    X4-X1              ↱X4:=ABS(X1)
          RJ     RUEDI               ↱USES X3,X4,X7,B2,B7
          PL     X1,I1
          SX7    1R-                 ↱Ξ-Ξ
          SB2    B2+1                ↱B2=NO. OF SPACES REQUIRED
          SA7    SIGN+B2             ↱STOREΞ-Ξ
I1        SB3    B3-B2               ↱B3=NO. OF PRECEDING BLANKS
          LE     B3,PNEXT            ↱EXPAND FIELD LENGTH
          BLANKS B3
PNEXT     FETCHOUT B2
          SB2    B2-1
          GT     B2,PNEXT
          FINIS

FREAL     SA7    RET0
          NG     X3,BAD
          SA5    FMAX
          IX5    X3-X5
          BB3    X5,BAD
          JP     *+2

EREAL     SA7    RET0
```

```
          SB3    -1
REAL      NG     X2,BAD
          ZR     X2,WRTFIN          RETURN IF 0 FIELDWIDTH
          SA5    LMAX
          IX5    X2-X5
          PL     X5,BAD
          OR     X1,UNDEF
          ID     X1,UNDEF
          NX1    B0,X1
          ZR     X1,PTZERO
          SB7    X2-9
          LT     B3,RE1
          SB7    X2-3
          SB7    B7-B3
RE1        LT    B7,STAR            *CHECKS EFORMAT,BEGINS CHECK ON FFORMAT
          SX6    B0
          PL     X1,R2
          SX6    1
R2        SA6    NEGA
          RJ     BISECT
          SA6    SIGN               *STORE ABS(EXPO)
          SX6    1R-                *Ξ-Ξ
          NG     X0,R3              *IF EXPO IS NEG, JUMP
          SX6    1R+                *Ξ+Ξ
R3        SA6    SIGN-1             *STORE Ξ+Ξ ORΞ-Ξ
          GE     B3,FFORMAT         *ELSE BEGIN EFORMAT
         SB7    X2-7               *B7=Q=NO. OF DIGITS TO BE OUTPUT
                                   *SIGN+B7 ->TO MOST SIG. DIGIT
          SX0    B7-15
          NG     X0,R4              *MAX. NO. OF DIGITS = 14
          SB7    14
R4        RJ     FILL               *X4: ABS(FRACTIONAL PART) B7:Q
                                   *OUT: B3=Q,DIGITS STORED FROM SIGN+1 - SIGN+2
                                   *X6=2 IF ROUNDED 9ΞS, OTHERWISE =1
          SSIGN                     *X3=0 IF VALUE≥0,=1 IF VALUE <0
          SB3    B3+6               *+-X.X   XE+-XX .B3 DIGITS,6 SPECIAL CHR.
          SB3    -B3                *
          SB3    X2+B3              *B3=NO. OF PRECEDING BLKS.
          SA6    NEGA               * SAVE POINTER TO LEAST SIG. DIGIT
          BLANKS B3
          FETCHOUT B2               *WRITES SIGN INDICATOR
          SB2    B2-1
          FETCHOUT B2               *WRITES MOST SIG. DIGIT
          SX6    1R.                *Ξ.Ξ
          WROUT  A6
          SA3    NEGA               * FETCH POINTER
          SB3    X3                 *B3 POINTS TO LEAST SIG. DIGIT
R6        SB2    B2-1
          FETCHOUT B2
          GT     B2,B3,R6
          EQ     PTEXPO

CHECK     MACRO
          SB3    X2
          SB3    A0-B3               *REQUIRED AMOUNT-FIELD LENGTH
          GT     B3,STAR            *FIELD LENGTH TOO SMALL
          ENDM

FFORMAT   SX6    B3
          SA6    FFSPEC             *SAVE FFORMAT SPEF.
          SA3    SIGN-1             *FETCH SIGN OF EXPO
          SX3    X3-45B             *X3=0 IF EXPO ≥0,=1 OTHERWISE
          SA1    SIGN
          ZR     X3,F1
          BX1    -X1                *X1=SIGNED EXPO
F1        SX0    X1-14
```

```
          PL      X0,STAR          ↱IS NO. TOO LARGE
          SB7     X1+B3
          SB7     B7+1             ↱IF POS,B2=NO. OF TRAILING BLKS,
          SB2     B7-14            ↱IF POS,B2=NO. OF TRAILING BLANKS
          LT      B2,F2
          SB7     14               ↱MAX. INFO = 14 DIGITS
F2        NZ      X3,F7            ↱JUMP IF EXPO <0
  ABS(VALUE)≥1:
          SA6     B7+2             ↱Ξ  X--XΞ
          EQ      B3,F4
          SA0     A0+1             ↱Ξ  X--X.X--XΞ
F3        LE      B2,F4
          SA0     A0+B2            ↱Ξ  X--X.X--X   Ξ 9A0=NO. OF CHAR. REQUIRED
F4        CHECK                    ↱IS THERE ENOUGH SPACE
          SX6     B2-1             ↱SAVE NO. OF TRAILING BLKS
          SA6     SAVEB2
          RJ      FILL             ↱B2=INDEX.B3=Q,X6=1 OR 2
          SX0     X6-1             ↱X0=1 IF ROUNDED .ΞS,=0 OTHERWISE
          ZR      X0,F5
          SA0     A0+1             ↱REQUIRES ONE MORE SPACE
          CHECK
F5        SSIGN                    ↱STORES THE SIGN INDICATOR
          SB3     A0-1             ↱B3=NO. OF INFO CHAR
          SA0     X2
          SB3     A0-B3            ↱NO. OF PRECEEDITG BLKS
          BLANKS  B3
          SA3     FFSPEC           ↱FETCH F-SPEF=NO. OF DIGITS AFTER THE Ξ.Ξ
          SB3     X3
          SA4     SAVEB2           ↱FETCH NO. OF TRAILING BLANKS
          NG      X4,F6
          SB7     X4+1              ↱ADD IN CORRECTION
          SB3     B3-B7
F6        FETCHOUT B2
          SB2     B2-1
          GT      B2,B3,F6         ↱WRITES DIGITS FOR Ξ.Ξ
          EQ      B3,WREND2        ↱EXIT IF F-SPEF =0
          SX6     1R.              ↱Ξ.Ξ
          WROUT   A6
WREND1    FETCHOUT B2
          SB2     B2-1
WREND     GT      B2,WREND1
          SA3     SAVEB2           ↱FETCH NO OF TRAILING BLANKS
          NG      X3,WREND2
          SB3     X3+1             ↱ADD IN CORRECTION
          BLANKS  B3
WREND2    FINIS
  ABS(VALUE)<1:
F7        NE      B3,F10
          LT      B7,PTZERO        ↱WRITE AS AN INTEGER
          RJ      FILL
          SA3     SIGN+1
          SX3     X3-33B
          ZR      X3,PTZERO
          SB3     X2-2
          BLANKS  B3
          SX6     1R
          SA3     NEGA
          ZR      X3,F8
          SX6     1R-
F8        WROUT   A6
          SX6     34B
          WROUT   A6
          FINIS
F10       SA0     B3+4             ↱Ξ  X.<B3>Ξ A0=SPACE REQUIREMENT
          CHECK
          SX6     B2-1
```

```
          SA6     SAVEB2          ↱SAVE NO. OF TRAILING BLKS - 1
          GE      B7,F11          ↱ELSE VALUE =0 TO SPEF.ACCURACY
          SA3     FFSPEC
          SX6     X3+1
          SA6     SIGN
          EQ      PASS
F11       RJ      FILL            ↱B2->MOST SIG. DIGIT
PASS      SB3     A0-1
          SA0     X2
          SB3     A0-B3
          BLANKS  B3
          SA3     NEGA
          SX6     1R              ↱Ξ Ξ
          ZR      X3,F12
          SX6     1R-             ↱Ξ-Ξ
F12       WROUT   A6              ↱OUTPUTS SIGN INDICATOR
          SX6     338             ↱Ξ0Ξ
          SA3     SIGN
          NZ      X3,F13
          SX6     348             ↱Ξ1Ξ
          SB2     B2-1
F13       WROUT   A6              ↱WRITE LEADING DIGIT, 0 OR 1
          SX6     1R.
          WROUT   A6              ↱WRITE Ξ.Ξ
          SA3     SIGN            ↱ABS(EXPO)
          SB3     X3-1            ↱B3=NO. OF ZEROS AFTER THE Ξ.Ξ
F14       LE      B3,WREND
          SX6     338
          WROUT   A6
          SB3     B3-1
          EQ      F14

WRTFIN    EQU       WREND2
```

```
WRITEOCT OK
          SB1     1
          BX6     X1
          SA6     SIGN            ↱SAVE VALUE
          SB2     X2              ↱ 0<X2≤136
          LX0     B1,X2           ↱ <-
          SB2     X0+B2           ↱B2:=3*X2
          SB3     X2-20           ↱IF ≥1 THEN IS NO OF PRECEDING BLKS.
          LE      B3,NOCT
          BLANKS  B3
          SA1     SIGN
          SB2     60
NOCT      SB2     B2-3
          AX1     B2,X1           ↱ ->
          MX0     57
          BX0     -X0*X1
          SX6     X0+33B
          WROUT   A6
          SA1     SIGN
          GT      B2,NOCT
          FINIS


TORF      DATA    LΞ TRUEFALSEΞ
BARG      DATA    LΞ <BAD ARG>Ξ
NDEF      DATA    LΞ <UNDEF>  Ξ
HALF      DATA    0.5
LMAX      DATA    137
FMAX      DATA    133
LOG2      DATA    0.30103
C1.0      DATA    1.0
```

```
C10       DATA    10.0

P         DATA    1.0E0
          DATA    1.0E1
          DATA    1.0E2
          DATA    1.0E3
          DATA    1.0E4
          DATA    1.0E5
          DATA    1.0E6
          DATA    1.0E7
          DATA    1.0E8
          DATA    1.0E9
          DATA    1.0E10
          DATA    1.0E11
          DATA    1.0E12
          DATA    1.0E13
          DATA    1.0E14
NEGA      BSS     2
SIGN      BSS     17
SAVEB2    BSS     1
FFSPEC    BSS      1

   END OF WRITE PROCEDURE


          EJECT
  POSYS.1  (BEFORE LOADING OF FIRST OVERLAY)
  ----------------------------------------------------------------

  UPON ENTRY, A0 CONTAINS THE FIELD LENGTH, WHICH IS ≥ 40000B.

POSYS.1   SB1       1               B1 IS USED AS A CONSTANT


  ACTIVATE RPV:
          SA1       RPVREQ
          BX6       X1
          SA6       B1
          WAIT      1
  TRANSFER PARAMETER INFORMATION TO THE PARAMETER TABLE:
          SB2       B1+B1
          SA1       B2
PAR1      ZR        X1,PAR4
          SA2       PARAMS
PAR2      BX0       X1-X2
          ZR        X0,PAR3
          SA2       A2+B2
          NZ        X2,PAR2

  ILLEGAL KEYWORD:
PARERR1   SX0       ERRMSG1
          RJ        MESS
          EQ        A8N.EX3

  LEGAL KEYWORD FOUND:
PAR3      SA3       A2+B1
          NZ        X3,PARERR2
          SA2       A1+B1           LOAD PARAMETER VALUE
          BX6       X2
          SA6       A3              STORE PARAMETER VALUE
          SA1       A1+2
          EQ        PAR1

  CURRENT KEYWORD HAS ALREADY BEEN USED:
PARERR2   SX0       ERRMSG2
```

```
          RJ        MESS
          EQ        ABN.EX3

  CHECK WHETHER ALL KEYWORDS AND PARAMETERS HAVE BEEN TAKEN CARE OF:
PAR4      SX1       A1-B2
          SA2       64B               NUMBER OF KEYWORDS AND PARAMETERS
          SX2       X2
          BX0       X1-X2
          NZ        X0,PARERR1

  OPEN  PASCAL:
          SA1       PASCAL
          SX6       102B              OPEN FOR READ
          SX0       3ROPE
          RJ        PPREQ

  IF [LOADPAR] ≠ 0 WE OPEN THE LOAD FILE, SKIPF PASCAL BY 2,
  SET B5 := DPAR - PPAR AND A1/X1 := LOADFILE FET ADDRESS.
  OTHERWISE WE SET B5 := 0 AND A1/X1 := PASCAL FET ADDRESS.
******  B5 RETAINS ITS VALUE THOUGHOUT POSYS.1  *******
          SA2       LOADPAR
          SA1       PASCAL
          SB5       0
          ZR        X2,POS1.L0
          SX3       2
          RJ        SKIPF
          SX0       3ROPE
          SB5       DPAR-PPAR
          BX6       X2
          SA6       LOADFILE          STORE LFN
          SA1       A6
          SX6       102B              OPEN FOR READ
          RJ        PPREQ

  READ THE HEADER:
  AT THIS POINT, A1/X1 = FET OF PASCAL OR OF LOADFILE.
POS1.L0   SX6       12B               READ BINARY
          SX0       3RCIO
          RJ        PPREQ             (IN = OUT = HEADER)

  IF B5 ≠ 0 WE CHECK THE HEADER:
          ZR        B5,POS1.L1
          SA1       LOADFILE+2        IN
          SA2       A1+B1             OUT
          IX0       X1-X2
          SX0       X0-ENTRYPT-1+DATE      X0 - LENGTH OF HEADER
          ZR        X0,POS1.L1

  BAD HEADER READ:
          SX0       LOADMSG
          RJ        MESS
          EQ        ABN.EX3

  CHECK WHETHER THE FIELD LENGTH IS SUFFICIENT:
POS1.L1   SA5       REQRDFL           NECESSARY FL
          SX6       A0                ACTUAL FL
          IX0       X6-X5
          PL        X0,POS1.L2

  FIELD LENGTH TOO SMALL:
          SB2       5
          RJ        BIN.DISP
          SA6       ERRMSG5+3
          SX0       ERRMSG5
          RJ        MESS
          EQ        ABN.EX3
```

```
  STORE FIELD LENGTH. CORRECT FET LIMITS.
POS1.L2  SA6      F.LGTH
         SA6      PASCAL+4
         SA6      PASCLGO+4
         SA6      LOADFILE+4       ALSO KEEPFET+4

  SET [IC0]. ISSUE THE DATE MESSAGE.
         SA1      DATE
         BX6      X1
         ZR       B5,POS1.L3
         SA6      DATEMSG2+1
         SX0      DATEMSG2
         EQ       POS1.L4
POS1.L3  SA6      DATEMSG1+2
         SX0      DATEMSG1
POS1.L4  SX6      BIGBUF
         SA6      IC0
         RJ       MESS

  ENQUIRE WHETHER ONE OF THE P OR D INPUT OPTIONS IS USED.
  ALTER THE INPUT FET ACCORDINGLY AND OPEN THE SPECIFIED FILE:
         SA1      PPAR+B5          X1 := NAME OF P OR D FILE
         SA2      IONOGO           X2 := =INPUT=
         SA3      INFET-1
         MX0      1
         LX0      52
         ZR       X1,POS1.L5       JUMP IF NO NAME SPECIFIED
         BX2      X2-X1
         BX6      X0+X3            MASK := 30B
         ZR       X2,POS1.L5A      JUMP IF =INPUT=
         EQ       POS1.L6
POS1.L5  BX1      X2               NAME := =INPUT=
POS1.L5A BX6      -X0*X3           MASK := 20B
POS1.L6  SA6      A3               STORE MASK
         BX6      X1
         SA6      A3+B1            PUT NAME IN FET
 OPEN THE SPECIFIED FILE
         SB7      A3
         CALL     OPEN

  OPEN THE OUTPUT FILE.
         SA1      LPAR+B5          X1 := ZERO OR NAME OF L OR OF R FILE
         NZ       X1,POS1.L7
         SA1      IONOGO+1         X1 := =OUTPUT=
         BX6      X1
         SA6      LPAR+B5          STORE =OUTPUT= IF NO NAME SPECIFIED
  IF [OPTPAR] = =FC*= AND B5 ≠ 0 THEN LISTING GOES ON FQCFILE:
POS1.L7  NZ       B5,POS1.L8       IGNORE OPT IF [LOADPAR] ≠ 0
         SA2      OPTPAR
         ZR       X2,POS1.L8
         MX0      18
         BX0      X2*X0
         LX0      18
         SX0      X0-3RFC*
         NZ       X0,POS1.L8       IF [OPTPAR] = =FC*= THEN
         SA1      FQCNAME          BEGIN X1 := =FQCFILE= .,
         SX6      B1                  FQCFILFL := 1
         SA6      FQCFILFL         END
POS1.L8  SB7      OUTFET-1
         BX6      X1
         SA6      B7+B1            STORE OUTPUT FILE=S NAME IN FET
         CALL     OPEN

  CONVERT AND SAVE THE LINE LIMIT PARAMETER:
POS1.L9  SA1      LLPAR            LINE LIMIT
```

```
            SX6     1000                DEFAULT VALUE
            ZR      X1,POS1.L13

            MX0     54
            SX6     B0
            EQ      POS1.L12
POS1.L10    SX0     ERRMSG4
            RJ      MESS
            EQ      ABN.EX3
POS1.L11    SX3     X2-1R+
            PL      X3,POS1.L10         CHECK FOR DIGIT
            SX2     X2-1R0
            NG      X2,POS1.L10         CHECK FOR DIGIT
            LX5     B1,X6
            LX6     3
            IX6     X6+X5
            IX6     X6+X2
POS1.L12    LX1     6
            BX2     -X0*X1
            NZ      X2,POS1.L11
POS1.L13    SA6     LLPAR

  IF B5 = 0 REWIND PASCLGO (PASCLGO MIGHT ALREADY EXIST) AND SET
  [LINECTR] := -1, OTHERWISE SET [LINECTR] := X6.
  UPDATE RECPOS. LOAD AND EXECUTE:
            NZ      B5,POS1.L14
            SA1     PASCLGO
            SX6     528                 REWIND
            SX0     3RCIO
            RJ      PPREQ
            SX6     -1
            SA6     LINECTR
            SA1     PASCAL
            EQ      POS1.L15
POS1.L14    SA6     LINECTR
            SA1     LOADFILE
POS1.L15    RJ      LOAD
            SX6     3
            SB6     RECPOS
            NZ      B5,*+1              FOR [EXECFLAG]
            SB3     B0
            SA2     EXTFLAGS
            ZR      X2,POS1.L16
            SA4     LIMCODE
            SA1     EXTVECT
            RJ      LOADEXTS
POS1.L16    SB2     B0                  NO FREQUENCY COUNTS FOR COMPILER
            RJ      GO
            EJECT
  POSYS.2  (AFTER EXECUTION OF FIRST OVERLAY)
------------------------------------------------------------------

  EMPTY THE OUTPUT BUFFER:
POSYS.2     SB7     OUTFET-1
            CALL    EOR

  TERMINATE IF LOADPAR ≠ 0:
            SA1     LOADPAR
            ZR      X1,POS2.L1
            SX0     ENDMSG
            RJ      MESS
            EQ      N.EXIT3

  ISSUE THE =END OF COMPILATION= MESSAGE:
  IF SYNTACTICAL ERRORS WERE FOUND, PRINT ERROR MESSAGES AND TAKE
  CARE OF FQCFILE:
```

```
POS2.L1  SX0      ENDCOMP
         RJ       MESS
         SA1      ERRFLAG
         ZR       X1,POS2.L2
         SX0      ERRMSG6
         RJ       MESS
         RJ       PRTERR
         SB7      OUTFET-1
         CALL     EOR
         RJ       PRTFQC
         EQ       N.EXIT2

  REWIND PASCLGO:
POS2.L2  SX6      1468              OPEN/REWIND FOR READ AND WRITE
         SA1      PASCLGO
         SX0      3ROPE
         RJ       PPREQ

  MOVE SOME OUTPUT PARAMETERS FROM PARMLIST TO THE HEADER
         SB2      5                 NUMBER OF PARAMETERS TO MOVE
         SB1      PARMLIST-PARMLIST-6
         BX6      X1
         SA6      A1+B3
         SB2      B2-B1
         NE       B2,*-1
         EJECT
  POSYS.3  (BEFORE EXECUTION OF COMPILED PROGRAM)
  ------------------------------------------------------------------

  IF A NEW EXECUTION FIELD LENGTH IS SPECIFIED WE CONVERT IT TO BINARY:
POSYS.3  SA1      FLPAR             EXECUTION FL PARAMETER
         SA2      F.LGTH            ACTUAL FIELD LENGTH
         BX6      X2
         ZR       X1,POS3.L4

         RJ       DISP.BIN            CONVERT TO BINARY
         PL       X6,POS3.L4
         SX0      ERRMSG3
         RJ       MESS
         EQ       ABN.EX2

  CHECK WHETHER THE USERES MAIN PROGRAM CAN BE LOADED:
POS3.L3  SX0      ERRMSG7
         RJ       MESS
         EQ       ABN.EX2
POS3.L4  SA6      FLPAR             SAVE EXECUTION FIELD LENGTH
         SA1      LIMCODE
         SX1      X1+JMPTABLG+1
         IX0      X6-X1             RESULT MUST BE ≥ 0
         NG       X0,POS3.L3
         SA1      LPJMPTAB
         SX1      X1+PMDLGTH
         IX0      X6-X1
         NG       X0,POS3.L3

  IF THE EXECUTION FIELD LENGTH IS DIFFERENT FROM THE ACTUAL ONE
  WE ISSUE A MEM REQUEST AND CORRECT THE FET LIMITS:
         SA1      FLPAR
         SA2      F.LGTH
         BX0      X2-X1
         ZR       X0,POS3.L6
         RJ       MEMREQ
         SA6      PASCLGO+4         LIMIT
         SA6      PASCAL+4
         SA6      KEEPFET+4
```

```
    IF OPTPAR = NOGO THEN IF KEEPFPAR = 0 THEN TERMINATE
                                          ELSE GOTO POS3.L14;
POS3.L6  SA1     OPTPAR
         SA2     IONOGO+2          X2 := ΞNOGOΞ
         BX0     X1-X2
         NZ      X0,POS3.L7
         SA1     KEEPFPAR
         ZR      X1,N.EXIT2
         EQ      POS3.L14

    TAKE ACTIONS FOR THE D FILE:
POS3.L7  SA3     IONOGO            X3 := ΞINPUTΞ
         SA1     DPAR              X1 := NAME OF DATA FILE
         NZ      X1,*+1
         BX1     X3                IF X1 = 0 THEN X1 := ΞINPUTΞ
         SA2     PPAR              X2 := NAME OF PROGRAM FILE
         BX0     X1-X2
         ZR      X0,POS3.L8        IF P FILE = D FILE WE DO NOTHING

    WE CALL OPEN FOR THE SPECIFIED FILE, THEREBY ALSO INITIALIZING ALL
    EFET POINTERS:
         BX6     X1
         SA6     INFET             CORRECT NAME IN FET
         SB7     A6-B1
         BX3     X1-X3
         MX0     1
         LX0     52
         SA2     B7
         BX6     -X0*X2            MASK := 20B
         ZR      X3,POS3.L7A       JUMP IF D=INPUT
         BX6     X0+X2             MASK := 30B
POS3.L7A SA6     B7                SET MASK
         CALL    OPEN              OPEN AND INITIALIZE POINTERS

    TAKE ACTIONS FOR THE RESULT FILE:
POS3.L8  SA1     RPAR              X1 := NAME OF RESULT FILE
         SB7     OUTFET-1
         NZ      X1,*+1
         BX6     X0NOGO+1          IF X1 = 0 THEN X1 := ΞOUTPUTΞ
         SA6     B7+B1             CORRECT NAME IN FET
         CALL    OPEN

    [LINECTR] := LINE LIMIT:
         SA1     LLPAR
         BX6     X1
         SA6     LINECTR

    LOAD THE USER PROGRAM:
POS3.L14 SA1     PASCLGO
         RJ      LOAD

    IF KEEPING IS WANTED THEN KEEP AND CHECK FOR NOGO:
         SA1     KEEPFPAR
         ZR      X1,POS3.L15
         RJ      KEEP
         SA1     OPTPAR
         SA2     IONOGO+2
         BX0     X1-X2
         ZR      X0,N.EXIT2

    PREPARE FREQUENCY COUNTING, IF DESIRED, AND EXECUTE
    THE LOADED PROGRAM:
POS3.L15 SA2     EXTFLAGS
         ZR      X2,POS3.L16
         SA4     LIMCODE
         SA1     EXTVECT
```

```
          RJ        LOADEXTS
POS3.L16  SB2       B0
          SA1       OPTPAR
          MX0       12
          BX0       X1*X0
          LX0       12
          SX0       X0-2RFC
          NZ        X0,POS3.L17
          RJ        CPFQC1
          SB2       1

          IF THE FIELD LENGTH IS < BIGBUF+CTRMAX+1 IT MUST BE RAISED:
          SA2       F.LGTH
          SX0       X2-BIGBUF-CTRMAX-1
          PL        X0,POS3.L17
          SX1       BIGBUF+CTRMAX+1
          RJ        MEMREQ
POS3.L17  SB3       B1+B1             FOR [EXECFLAG]
          RJ        GO

  CALL CPFQC2 IF NECESSARY:
          ZR        B2,POSYS.4

  EMPTY THE OUTPUT BUFFER:
          SB7       OUTFET-1
          CALL      EOR

  PRINT OUT FREQUENCY COUNTS:
          SB7       OUTFET-1
          SA1       LPAR
          BX6       X1
          SA6       OUTFET
          CALL      OPEN
          RJ        CPFQC2

          SX6       B0
          SA6       FQCFILFL
          SA1       FQCNAME
          SA3       PASCLGO           SAVE FILE NAME
          BX6       X1
          SA6       A3
          SX0       3RCLO
          SX6       1728              CLOSE/UNLOAD
          SA1       A3
          RJ        PPREQ
          BX6       X3
          SA6       A3                RESTORE
          EJECT
    POSYS.4  (TERMINATION)
    ------------------------------------------------------------

POSYS.4   SX0       ENDMSG
          RJ        MESS

  NORMAL EXITS:
      EMPTY OUTBUF:
N.EXIT1   SB7       OUTFET-1
          CALL      EOR

      CLOSE / UNLOAD PASCLGO AND PMDFILE:
N.EXIT2   SX0       3RCLO
          SA1       PASCLGO
          SX6       1728              CLOSE/UNLOAD
          RJ        PPREQ
          SX0       3RCLO
          SA1       PMDFET
```

```
          SX6       172B
          RJ        PPREQ

      ISSUE END REQUEST:
N.EXIT3   SX6       3REND
          LX6       42
          SA6       B1
          PS

  ABNORMAL EXITS:

RPVTAB    DATA      0
          BSS       16

  ENTRY HERE AFTER ERROR DETECTED BY SCOPE:
  FIND ADDRESS. WRITE MODE IF MODE ERROR:
          SB1       1
          SA5       RPVTAB
          SX1       X5-2              X1 := ERROR CODE - 2
          LX5       24                ADDRESS FROM XJ PACKAGE
          NZ        X1,F.AD0          CHECK FOR MODE ERROR
          SA5       B0                GET (RA+0)
          LX5       12
          SX6       X5                X6 := MODE
          SA1       MODEMSG
          IX6       X6+X1
          SA6       A1
          SX0       A1
          RJ        MESS              ISSUE MODE MESSAGE
          LX5       18                X5 := ADDRESS (FROM RA+0)

  IF DEATH OCCURRED DURING EXECUTION OF A USER PROGRAM WE WANT
  AN ADDRESS POINTING INTO IT, OTHERWISE THE TRUE ADDRESS:
F.AD0     SA1       EXECFLAG
          ZR        X1,ABN.EX11
F.AD1     SB2       X5
          SA2       LIMCODE
          SB7       X2
          LT        B2,B7,*+2         EXTERNAL ROUTINES
          SA5       B6+B1
          BX5       B2AD4
          SX5       B2
          SB7       BIGBUF
          GE        B2,B7,F.AD4       USER PROGRAM
          SB7       POSYS.4
          LT        B2,B7,F.AD2       TERMINATION IF POSYS.4
          JP        B2
F.AD2     SA5       RET0
          SB7       OPEN
          GE        B2,B7,F.AD4       OPEN, ... , WRITE
          SA5       RET1
          SB7       GETC
          GE        B2,B7,F.AD1       GETC, PUTC, EOR
          SA5       RET2
          SB7       GETB
          GE        B2,B7,F.AD1       GETB,PUTB
          SA5       PPREQ
          LX5       30
          SB7       A5
          GE        B2,B7,F.AD1       PPREQ
          SB7       NULL1
          LT        B2,B7,F.AD3
          JP        B2                ERROR EXIT TRANSFER WORDS
F.AD3     BX5       X7                OTHER TRANSFER WORDS
F.AD4     EQ        ABN.EX11
```

```
      ENTRY HERE AFTER ERROR DETECTED BY PASCAL:
      DEACTIVATE RPV:
ABN.EX1  SB1      1
         SA1      RPVREQ
         SX6      B0
         SA6      RPVTAB
         BX5      X7
         MX2      3
         LX2      27
         BX6      -X2*X1
         SA6      B1
         WAIT     1
         RJ       MESS

ABN.EX11 SX5      X5-1
         SX7      B5
         SA7      SAVEB5          IN CALLPMD
         BX7      X5
         SA7      SAVEX5          IN CALLPMD
         SB2      6
         RJ       BIN.DISP
         SA6      ADDRMSG+1
         SX0      ADDRMSG
         RJ       MESS
         SX1      B0              EOL
         SX2      B1
         CALL     CHAR
         SX0      3RCIO
         SA1      OUTFET
         SX6      268
         RJ       PPREQ
         SA1      EXECFLAG
         SB2      X1-2
         NZ       B2,ABN.EX2
         SX6      B0
         SA6      A1
         SX6      -B1
         SA6      LINECTR
         RJ       CALLPMD

      IF [LOADPAR] = 0 CLOSE/UNLOAD PASCLGO AND PMDFILE:
ABN.EX2  SA1      LOADPAR
         NZ       X1,ABN.EX21
         SX0      3RCLO
         SA1      PASCLGO
         SX6      1728
         RJ       PPREQ
         SX0      3RCLO
         SA1      PMDFET
         SX6      1728
         RJ       PPREQ

      TAKE CARE OF FQCFILE:
ABN.EX21 RJ       PRTFQC


           ISSUE ABORT/NODUMP REQUEST:
ABN.EX3  SA1      ABTREQ
         BX6      X1
         SA6      B1
         PS
         EJECT
HEADER   BSS      0

DATE     BSS      1               DATE OF BIRTH OF BINARY PROGRAM
REQRDFL  BSS      1               REQUIRED FIELD LENGTH
```

```
LOADPT   BSS      1                LOAD POINT OF BINARY INFORMATION
LIMCODE  BSS      1                LIMIT OF CODE
LPJMPTAB BSS      1                LOAD POINT OF JUMP TABLE
EXTFLAGS BSS      1                EXTERNAL ROUTINES FLAGS
ENTRYPT  BSS      1                ENTRY POINT

OUTBUF   EQU      *
INBUF    EQU      OUTBUF+BUFLGTH

 BIGBUF  EQU      5000B

         END
        (*************************************************
         *                                               *
         *         P A S C A L   C O M P I L E R         *
         *         *****************************         *
         *                                               *
         *                  23/11/72                     *
         *                                               *
         *  AUTHORS:  U. AMMANN,  R. SCHILD              *
         *                                               *
         *            FACHGRUPPE COMPUTERWISSENSCHAFTEN  *
         *            EIDG. TECHNISCHE HOCHSCHULE        *
         *    CH-8006 ZUERICH                            *
         *                                               *
         *************************************************)

        (*$T-,O+,R+   COMPILE WITH STACK OVERFLOW TEST AND
                      ROUNDING INSTRUCTIONS*)

  CONST  NOP = 460000B, NILVAL = 2000000B,
     JMPCST = 6165000000400000000B, EQJMP = 0400000000460004600 0B,
     MAX10 = 3146314631463146313B, MASK = -777777B,
     MAXLABS = 20, MAXEXLABS = 10, JMPMAX = 49,
     CSTMAX = 40, CODMAX =  1000, UNDMAX = 70,
     DISPLIMIT = 20, PTLIMIT = 10, FILLIMIT = 10,
     MAXLEVEL = 7,
     (*DON=T INCREASE WITHOUT INCREASING UPPER BOUND OF RG3 TOO*)
     WORDLENGTH = 60,

     PROCDR = -1,  OBJ = +1,  (*FOR PMDFILE*)

     GETC = 101B, PUTC = 103B, GETB = 113B, PUTB = 102B,
  (*ADDRESSES OF RESIDENT STANDARD ROUTINES*)

     ASSERR = 170B, INXERR = 171B, DIVERR = 172B, STOFLERR = 174B,
  (*RUNTIME ERROR EXITS*)

     INPT = 212B, OUTPT = 233B;
  (* ADDRESSES OF STANDARD FILES INPUT AND OUTPUT *)

  TYPE CTP = ^CONTEXTTABLE;
     CWORD = ARRAY [1..10] OF CHAR;
     ADDRESS = 0..1777778 ;
     SHRTINT = -377777B .. +377777B;
     BITRANGE = 0 .. 60;  RG3 = 0 .. 7;
     ATTRKIND = (VARBL,SVAL,LVAL,LCOND);
     ATTR = RECORD TYPTR : CTP ;
               CASE KIND : ATTRKIND OF
                  VARBL: (ACCESS : (DRCT,INDRCT,INXD) ;
                          BREG : RG3 ; DPLMT : SHRTINT ;
                          CASE PCKD : BOOLEAN OF
                             FALSE: ;
                             TRUE: (BITADR, BITSZ : BITRANGE;
                                    MASK: BOOLEAN)) ;
                  SVAL:  (VAL : SHRTINT) ;
```

```
                LVAL:  (CTERM : SHRTINT) ;
                LCOND: (JMP : 0 .. 3 ; ARITH : BOOLEAN)
          END ;
    *DESCRIBES EXPRESSIONS TO BE COMPILED.     EXAMPLES:
        VARBL:          DRCT:                  I                **)
                        INDRCT:                INPUT↑            *)
                        INXD:                  A[K]              *)
        SVAL:                                  4                **)
        LVAL:                                  2*I - 1           *)
        LCOND:          ARITH:                 X ≥ 4.1           *)
                       ¬ARITH:                 B < TRUE          *)

        *) RESULT OF CODEGENERATION IN REGISTER X-RP
       **) NO CODE GENERATION UNTIL NOW↓


    OPTPWR = (NOOPT,PUREP,POSP,NEGP);
    IDCLASS = (TYPES,KONST,PROC,VARS,FIELD,TAGFIELD,DUMMYCLASS);
    TYPFORM = (NUMERIC,SYMBOLIC,POINTER,POWER,
               ARRAYS,RECORDS,CLASSS,FILES);
    IDKINDS = (ACTUAL,FORMAL);
    WHERE = (BLOCK,CWITH,VWITH);
    PKINDS = PROCDR .. OBJ; *FOR PMDFILE↓

VAR PARMLIST : RECORD    IC0 : ADDRESS ;
                         ERRFLAG : BOOLEAN ;
                         ERRNRS : ARRAY [0..3] OF SET OF 0..31 ;
                         LOADPT : ADDRESS ;
                         LIMCODE : ADDRESS ;
                         LPJMPTAB : ADDRESS ;
                         EXTFLAGS : SET OF 0..4 ;
                         ENTRYPT : ADDRESS
                END ;
    JMPTAB : ARRAY [0..JMPMAX] OF INTEGER;
    JMPIX : SHRTINT;
    *TRANSFER VECTOR FOR CALLS OF FORWARD DECLARED PROCEDURES AND
     GOTO STATEMENTS LEADING OUT OF PROCEDURES↓

    CSTTB : ARRAY [1..CSTMAX] OF
       RECORD VALU : INTEGER; INX : SHRTINT  END;
    LCX : SHRTINT;
    *CONTAINS ALL CONSTANTS C, ABS(ORD(C)) ≥ 2**17, OCCURRING IN THE
     PROCEDURE ACTUALLY BEING COMPILED TOGETHER WITH AN INDEX INTO
     UNDLAB WHERE THEIR OCCURRENCES IN THE CODE OF THIS PROCEDURE
     ARE CHAINED↓

    PILEV,FILEV : ARRAY [0..MAXLEVEL] OF SHRTINT;
    PFL : ARRAY[0..FILLIMIT] OF ADDRESS;
    FILPTS : ARRAY [0..FILLIMIT] OF
       PACKED RECORD PTR : CTP;  SPEC : SHRTINT  END;
    PFTOP,FILTOP : SHRTINT;
    *PFL CONTAINS THE ADDRESSES OF LOCAL CLASSES,
       PILEV CONTAINS POINTERS INTO PFL, PFTOP = TOP OF PFL.
     FILPTS CONTAINS POINTERS TO ENTRIES FOR LOCAL FILES,
       FILEV CONTAINS POINTERS INTO FILPTS, FILTOP = TOP OF FILPTS↓

    EXTAB : ARRAY [1..MAXEXLABS] OF
       PACKED RECORD EXVAL,JMPTABIX : SHRTINT  END;
    CEXTABIX : SHRTINT;
    *CONTAINS THE EXPLICITLY DECLARED LABELS OF ALL PROCEDURES
     NOT YET CLOSED, TOGETHER WITH THEIR CORRESPONDING INDEX
     INTO JMPTAB↓

    LABTAB : ARRAY [1..MAXLABS] OF
       PACKED RECORD LABVAL,FLD2,FLD3 : SHRTINT  END;
    CLABIX : SHRTINT;
```

```
↱CONTAINS ALL LABELS MET SO FAR IN THE BODY OF THE PROCEDURE
 ACTUALLY BEING COMPILED, TOGETHER WITH INFORMATION WHETHER
 LABEL DEFINITION (<LABEL>:) ALREADY FOUND OR NOT (FLD2).
 IN THE FORMER CASE FLD3 CONTAINS THE CORRESPONDING ADDRESS,
 WHERE IN THE LATTER CASE FLD3 CONTAINS AN INDEX INTO UNDLAB
 WHERE THE OCCURRENCES ARE CHAINED↓

UNDLAB : ARRAY [1..UNDMAX] OF
   PACKED RECORD SUCC, PLACE : SHRTINT ;
            LFTSH : BITRANGE ;
         END ;
CHNIX : SHRTINT;
↱ACTS AS LISTSTRUCTURE, CHAINING OCCURRENCES OF CONSTANTS AND
 JUMPS TO NOT YET REACHED LABELS IN THE CODE OF THE PROCEDURE
 ACTUALLY BEING COMPILED.
 CHNIX = HEAD OF FREE LIST↓

PTLIST : ARRAY [0..PTLIMIT] OF
   RECORD HNAME : ALFA;  PPTR : CTP  END;
PTX : SHRTINT;
↱PTLIST CONTAINS NAMES OF YET UNDECLARED CLASSES AND
   POINTERS TO THE CORRESPONDING POINTER-TYPE ENTRIES
 PTX = TOP OF PTLIST ↓

ERRLIST : ARRAY [1..10] OF
   PACKED RECORD POS,NMR : SHRTINT  END;
ERRINX,CHCNT : SHRTINT;  EOLFLAG,ERR : BOOLEAN;
↱ERRLIST CONTAINS THE POSITIONS AND NUMBERS OF THE ERRORS
   ON ONE LINE
   ERRINX = TOP OF ERRLIST
   CHCNT = POSITION OF LAST READ CHARACTER
 ERRLIST,ERRINX,CHCNT,EOLFLAG ARE USED BY NEXTCH
   AND ERROR.
 ERR IS SET WHENEVER ERROR HAS BEEN CALLED AND IS TESTED
   (AND RESET) BY SEVERAL PROCEDURES ↓

DISPLAY : ARRAY [0..DISPLIMIT] OF
PACKED RECORD FNAME : CTP;
      CASE OCCUR : WHERE OF
         BLOCK : ;
         CWITH : (CDSPL : SHRTINT; CLEV : RG3);
         VWITH : (VDSPL : SHRTINT);
END;
TOP,DISX : SHRTINT;
↱TOP = TOP OF DISPLAY, DISX IS RETURNED BY SEARCH↓

CODE : ARRAY [0..CODMAX] OF INTEGER;
CA,CP : SHRTINT;  BUF : INTEGER;
PASCLGO[OUT] : FILE OF INTEGER;
↱CODE IS WRITTEN ONTO PASCLGO.
 BUF IS THE CURRENT ELEMENT OF CODE.
 CA = INDEX TO CODE,  CP = NUMBER OF QUARTER-WORDS USED ↓
LASTOP, LASTI : SHRTINT ;   ↱TO AVOID B XI XJ - INSTRUCTIONS↓

PMDFILE[OUT]: FILE 1 OF
   PACKED RECORD PNAME : ALFA;
      CASE PKIND : PKINDS OF
      OBJ: (ACCESS,TYP,RLA : SHRTINT);
      PROCDR: (LINK,SA,COUNT : SHRTINT);
   END;
PMCTR,LASTLINK : SHRTINT;

LC,IC : ADDRESS;
↱LOCATION COUNTER AND INSTRUCTION COUNTER↓

LEVEL,RP,RP1 : RG3;
```

```
      *RP,RP1 = REGISTER-POINTERS (STACK OF X-REGS)↓
      DP : BOOLEAN ;
      *DP = TRUE : DECLARATION PART (USED BY NEXTCH)↓

      PRCODE,ASSCHECK,INXCHECK,DIVCHECK,STOFLCHECK,
      ROUNDING : BOOLEAN ;    *COMPILER OPTIONS↓

      GATTR : ATTR;    *GLOBAL ATTRIBUTE RECORD ↓

      CONTEXTTABLE : CLASS 450 OF
      PACKED RECORD
      NAME : ALFA;  NXTEL : CTP;
         CASE KLASS : IDCLASS OF
         TYPES : (SIZE : SHRTINT;
            CASE FORM : TYPFORM OF
            NUMERIC : (BITS : BITRANGE; MIN,MAX : INTEGER);
            SYMBOLIC : (FCONST,PWSET : CTP; BITSIZE : BITRANGE);
            POINTER : (DOMAIN,ELTYPE : CTP);
            POWER : (ELSET : CTP; PWBITS : BITRANGE);
            ARRAYS : (AELTYPE,INXTYPE : CTP;  LO,HI : SHRTINT;
               OPTTYP : OPTPWR; EXP1,EXP2 : BITRANGE);
            RECORDS : (FSTFLD,RECVAR : CTP);
            CLASSS : (PELTYPE : CTP);
            FILES : (FELTYPE : CTP)    );
         KONST : (CONTYPE, SUCC : CTP; VALUES : INTEGER);
         PROC : (PROCTYPE,FORMALS : CTP;
            PROCKIND : IDKINDS;
            PROCADDR : ADDRESS; PROCLEVEL : RG3;
            SEGSIZE : SHRTINT);
         VARS : (VTYPE : CTP; VKIND : IDKINDS;
            VADDR : ADDRESS; VLEVEL : RG3);
         FIELD : (FLDTYPE : CTP; FLDADDR : ADDRESS;
            BITDISPL,BITWIDTH : BITRANGE);
         TAGFIELD : (CASESIZE : SHRTINT; VARIANTS : CTP;
            CASE TAGVAL : BOOLEAN OF
            FALSE : (CASETYPE : CTP);
            TRUE : (CASEVAL : INTEGER)    );
      END;

      INTPTR,REALPTR,ALFAPTR,CHARPTR,BOOLPTR,PREDEFP,NILPTR,
         UNDECPTR,PNUMPTR,EXTPTR,LAMPTR : CTP; *CONSTANT POINTERS↓

      NEXT,CTPTR,MAXCTP : CTP;
      *VARIABLES POINTING INTO CONTEXTTABLE↓

      CH : CHAR; AVAL : ALFA; IVAL : INTEGER; A : CWORD;
      NO,CL : SHRTINT;
      *CH IS OUTPUT BY NEXTCH AND USED BY INSYMBOL.
         AVAL/IVAL, NO, CL  ARE OUTPUT BY INSYMBOL↓

      TCT,TMAX,BODPL,KK : SHRTINT ;
      PT : CTP; IT,IT1 : INTEGER;
   *AUXILIARY VARIABLES, USED IN MAIN PGM AND SEVERAL PROCEDURES↓

      DIGITS : SET OF CHAR ;
      *0: SIN/COS, 1: EXP, 2: LN, 3: SQRT, 4: ARCTAN↓

      *CONSTANTS AND CONSTANT TABLES: ↓
      TWOTO17 : INTEGER;
      SPLITSTAT : ARRAY [0..48] OF SHRTINT;
      ERRCL,TERRCL : ARRAY [0..48] OF (IRRELSY,BEGSY,ENDSY);
      BLANK : ALFA;
      WD : ARRAY [0..32] OF ALFA;
      WNO,WCL : ARRAY [0..32] OF SHRTINT;
      WL : ARRAY [0..11] OF SHRTINT;
      * WD = WORD DELIMITERS
```

```
      WNO,WCL = CORRESPONDING VALUES OF  NO  AND  CL
      WORD DELIMITERS OF LENGTH  I  START IN WD[WL[I]]  ↓
    SYMNO,SYMCL : ARRAY [Ξ+Ξ..Ξ;Ξ] OF SHRTINT;
    ↱NO AND CL OF EACH CHARACTER↓
    JB1,JB2,JX1,JX2,REL,FS,R1 : ARRAY [0..7] OF CHAR;
    SH : ARRAY [12..23] OF CHAR;
    ↱ARRAYS JB1 ... SH ARE USED BY PRTCOMP↓
    INITNAM : ARRAY [-1..37] OF ALFA;
    ↱CONTAINS THE PREDEFINED IDENTIFIERS,
      USED FOR INITIALIZING THE CONTEXT TABLE↓

↱ ---------------------------------------------------------------- ↓

↱ VARIABLE INITIALIZATIONS  ↓
  VALUE
    TWOTO17 = 400000B ;
    SPLITSTAT = (1,2,19*1,3,1,4,1,1,5,1,6,1,7,1,8,2*1,9,12*1,10) ;
    ERRCL = (16*IRRELSY,ENDSY,4*IRRELSY,BEGSY,ENDSY,BEGSY,IRRELSY,
            ENDSY,BEGSY,IRRELSY,BEGSY,ENDSY,BEGSY,IRRELSY,BEGSY,
            2*IRRELSY,BEGSY,IRRELSY,ENDSY,2*IRRELSY,2*ENDSY,
            IRRELSY,3*ENDSY,2*IRRELSY,BEGSY);
    TERRCL = (9*IRRELSY,BEGSY,ENDSY,IRRELSY,ENDSY,3*IRRELSY,
      ENDSY,IRRELSY,BEGSY,2*IRRELSY,3*ENDSY,2*IRRELSY,ENDSY,
      IRRELSY,ENDSY,IRRELSY,ENDSY,IRRELSY,ENDSY,2*IRRELSY,
      ENDSY,IRRELSY,ENDSY,BEGSY,IRRELSY,2*ENDSY,IRRELSY,
      3*ENDSY,2*IRRELSY,ENDSY);
    BLANK = Ξ  Ξ ;
    WD = (ΞIFΞ,ΞDOΞ,ΞTOΞ,ΞOFΞ,ΞINΞ,
      ΞENDΞ,ΞNILΞ,ΞFORΞ,ΞDIVΞ,ΞMODΞ,ΞVARΞ,ΞSETΞ,
      ΞTHENΞ,ΞELSEΞ,ΞGOTOΞ,ΞCASEΞ,ΞWITHΞ,ΞTYPEΞ,ΞFILEΞ,
      ΞBEGINΞ,ΞUNTILΞ,ΞWHILEΞ,ΞARRAYΞ,ΞVALUEΞ,ΞCLASSΞ,
        ΞCONSTΞ,ΞLABELΞ,
      ΞREPEATΞ,ΞDOWNTOΞ,ΞRECORDΞ,ΞPACKEDΞ,
      ΞFUNCTIONΞ,ΞPROCEDUREΞ);
    WNO = (23,31,33,27,8,
      22,36,32,6,6,43,38,
      24,25,35,26,43,37,38,
      21,29,30,38,47,38,41,40,
      28,33,38,42,
      44,45);
    WCL = (0,0,1,0,7,
      0,0,0,4,5,0,5,
      6*0,3,
      0,0,0,1,0,4,0,0,
      0,2,2,0,
      0,0);
    WL = (0,0,0,5,12,19,27,31,31,32,33,33);
    SYMNO = (7,7,6,6,9,10,4,8,0,15,17,0,11,12,
      19,8,13,7,6,18,14,8,8,8,8,5,16);
    SYMCL = (1,2,1,2,0,0,1,6,3,0,0,2,0,0,
      0,5,0,3,3,0,0,1,4,2,3,1,0);
    JB1 = (ΞPΞ,ΞRΞ,ΞJΞ,Ξ Ξ,ΞEΞ,ΞNΞ,ΞGΞ,ΞLΞ);
    JB2 = (ΞSΞ,ΞJΞ,ΞPΞ,Ξ Ξ,ΞQΞ,ΞEΞ,ΞEΞ,ΞTΞ);
    JX1 = (ΞZΞ,ΞNΞ,ΞPΞ,ΞNΞ,ΞIΞ,ΞOΞ,ΞDΞ,ΞIΞ);
    JX2 = (ΞRΞ,ΞZΞ,ΞLΞ,ΞGΞ,ΞRΞ,ΞRΞ,ΞFΞ,ΞDΞ);
    REL = (Ξ Ξ,Ξ^Ξ,Ξ∨Ξ,Ξ-Ξ,Ξ=Ξ,Ξ≠Ξ,Ξ≥Ξ,Ξ<Ξ);
    FS  = (ΞFΞ,ΞRΞ,ΞDΞ,Ξ Ξ,Ξ Ξ,ΞAΞ,ΞBΞ,ΞXΞ);
    R1  = (ΞAΞ,ΞBΞ,ΞXΞ,ΞXΞ,ΞAΞ,ΞAΞ,ΞBΞ,ΞBΞ);
    SH  = (ΞFΞ,ΞDΞ,ΞRΞ,ΞIΞ,ΞLΞ,ΞAΞ,ΞLΞ,ΞAΞ,ΞNΞ,ΞZΞ,ΞUΞ,ΞPΞ);
    INITNAM = (ΞWEORΞ,ΞGETΞ,ΞPUTΞ,ΞRESETΞ,ΞREWRITEΞ,
      ΞPACKΞ,ΞUNPACKΞ,ΞINSERTΞ,ΞAPPENDΞ,
      ΞREADΞ,ΞWRITEΞ,ΞNEWΞ,
      ΞODDΞ,ΞORDΞ,ΞCHRΞ,ΞEOFΞ,ΞABSΞ,ΞSQRΞ,
      ΞTRUNCΞ,ΞPREDΞ,ΞSUCCΞ,ΞEOLΞ,ΞALFALENGΞ,
      ΞINPUTΞ,ΞOUTPUTΞ,ΞALFAΞ,ΞREALΞ,ΞCHARΞ,ΞBOOLEANΞ,
      ΞFALSEΞ,ΞTRUEΞ,ΞINTEGERΞ,Ξ  Ξ,
```

```
      ΞSINΞ,ΞCOSΞ,ΞEXPΞ,ΞLNΞ,ΞSQRTΞ,ΞARCTANΞ) ;

  ↱TWO STANDARD PROCEDURES  INSERT  AND  APPEND  ARE PROVIDED FOR
    CODE GENERATION.
   APPEND(A,B,C) SHIFTS THE CONTENTS OF A LEFT B BITS AND
    ΞORΞS C INTO IT. RESULT IN A. B,C UNCHANGED.
   INSERT(A,B,C) SHIFTS THE CONTENTS OF A LEFT B BITS AND
    ΞORΞS THEM INTO C. RESULT IN C. A,B UNCHANGED
   A,B,C ARE OF TYPE INTEGER ↳


  PROCEDURE PRTERR ;
    VAR LASTPOS,FREEPOS,CURRPOS,CURRNMR,F,K : SHRTINT ;
  BEGIN WRITE(Ξ ****  Ξ) ;
    LASTPOS := 0 ; FREEPOS := 1 ;
    FOR K := 1 TO ERRINX DO
    BEGIN WITH ERRLIST[K] DO
      BEGIN CURRPOS := POS ; CURRNMR := NMR END ;
      IF CURRPOS = LASTPOS THEN WRITE(Ξ,Ξ) ELSE
      BEGIN WHILE FREEPOS < CURRPOS DO
        BEGIN WRITE(Ξ Ξ) ;
          FREEPOS := FREEPOS + 1
        END ;
        WRITE(Ξ↑Ξ) ;
        LASTPOS := CURRPOS
      END ;
      IF CURRNMR < 10 THEN F := 1 ELSE
      IF CURRNMR < 100 THEN F := 2 ELSE F := 3 ;
      WRITE(CURRNMR : F) ;
      FREEPOS := FREEPOS + F + 1 ;
    END ;
    WRITE(EOL) ; ERRINX := 0
  END ↱PRTERR↳ ;

  PROCEDURE NEXTCH;
    ↱READS NEXT CHAR OF FILE INPUT.
      CONVERTS EOL TO BLANK,
      PRINTS ERROR SUMMARY AFTER EACH LINE (IF ERRORS PRESENT)
      AND PRINTS ADDRESS AT BEGINNING OF LINE ↳
  BEGIN
    IF EOLFLAG THEN
    BEGIN IF ERRINX > 0 THEN ↱PRINT ERRORS↳ PRTERR;
      EOLFLAG := FALSE;  CHCNT := 0;
      OUTPUT↑ := Ξ Ξ; PUT(OUTPUT);
      IF DP THEN WRITE(LC : 6 OCT) ELSE WRITE(IC : 6 OCT) ;
      OUTPUT↑ := Ξ Ξ;  PUT(OUTPUT);
    END ↱EOLFLAG↳;
    GET(INPUT); CH := INPUT↑;
    OUTPUT↑ := CH; PUT(OUTPUT); CHCNT := CHCNT + 1;
    IF CH = EOL THEN
    BEGIN CH := Ξ Ξ;  EOLFLAG := TRUE  END;
  END ↱NEXTCH↳;

  PROCEDURE ERROR(I:SHRTINT);
  BEGIN ERR := TRUE ;
    WITH PARMLIST DO
    BEGIN ERRFLAG := TRUE ;
      ERRNRS[I DIV 32] := ERRNRS[I DIV 32]∨[I MOD 32]
    END ;
    IF ERRINX ≥ 9 THEN
    BEGIN ERRLIST[10].NMR := 104 ; ERRINX := 10 ;
      WITH PARMLIST DO
      ERRNRS[3] := ERRNRS[3]∨[8] ;
    END ELSE
    BEGIN ERRINX := ERRINX + 1 ;
      ERRLIST[ERRINX].NMR := I
```

```
      END ;
      ERRLIST[ERRINX].POS := CHCNT
    END (*ERROR*);


  (* ------------------------------------------------------------------ *)

  (*  I N S Y M B O L              READS ONE PASCAL-SYMBOL  *)
    (* LIST OF SYMBOL NUMBERS
    NO   CL   SYMBOL              NO   CL   SYMBOL

     1        ID                  20        :=
     2    1   INTEGER CONST       21        BEGIN
     2    2   REAL    CONST       22        END
     2    3   ALFA    CONST       23        IF
     2    4   CHAR    CONST       24        THEN
                                  25        ELSE
                                  26        CASE
     5    1   ¬                   27        OF
     6    1   *                   28        REPEAT
     6    2   /                   29        UNTIL
     6    3   ^                   30        WHILE
     6    4   DIV                 31        DO
     6    5   MOD                 32        FOR
     7    1   +                   33   1    TO
     7    2   -                   33   2    DOWNTO
     7    3   v                   35        GOTO
     8    1   <                   36        NIL
     8    2   ≤                   37        TYPE
     8    3   ≥                   38   1    ARRAY
     8    4   >                   38   2    RECORD
     8    5   ≠                   38   3    FILE
     8    6   =                   38   4    CLASS
     8    7   IN                  38   5    SET
     9        (                   40        LABEL
    10        )                   41        CONST
    11        [                   42        PACKED
    12        ]                   43        VAR
    15        ,                   44        FUNCTION
    16        ;                   45        PROCEDURE
    17        .                   47        VALUE
    18        ↑                   48        WITH
    19        :

       NOT PASCAL SYMBOLS : BLANK  Ξ  (*  $  *)

    PROCEDURE INSYMBOL;
       VAR  SCALE,EXP : SHRTINT;  R,FAC,RVAL : REAL;
          DIGIT : ARRAY[0..19] OF SHRTINT;
                  (*FOR WORDLENGTH DIV 3 = 20 DIGITS*)
          I,K : SHRTINT;  BT1,SIGN : BOOLEAN;
          AA : ALFA; CH1 : CHAR;

    BEGIN
  1:   WHILE CH=Ξ Ξ DO NEXTCH;
       IF CH ≤ ΞZΞ THEN
       BEGIN  (* READ IDENTIFIER OR WORD DELIMITER *)
          K := 0;
          REPEAT IF K < ALFALENG THEN
             BEGIN K := K + 1;  A[K] := CH END;
             NEXTCH;
          UNTIL CH >Ξ9Ξ;
          IF K ≥ KK THEN KK := K ELSE
          REPEAT A[KK] := Ξ Ξ ; KK := KK - 1 UNTIL KK = K ;
          PACK(A,1,AA);
          (* TEST FOR WORD DELIMITER *)
```

```
         FOR I := WL[K] TO WL[K+1]-1 DO
            IF AA = WD[I] THEN
            BEGIN NO := WNO[I];  CL := WCL[I];
               IVAL := 0;  GOTO 2;
            END;
         ↑ IDENTIFIER ↓
         NO := 1;  CL := K;  AVAL := AA;
2:    END  ↑LETTER↓  ELSE
      IF CH ≤ ΞΞ THEN
      BEGIN NO := 2; CL := 1; I := 0;
         WHILE CH IN DIGITS DO
         BEGIN IF I < 20 ↑20 = WORDLENGTH DIV 3↓ THEN
            DIGIT[I] := ORD(CH) - ORD(Ξ0Ξ);
            I := I + 1; NEXTCH
         END;
         IVAL := 0;
         IF CH = ΞBΞ THEN ↑OCTAL↓
         BEGIN IF I > 20 THEN ERROR(2) ELSE
            FOR K := 0 TO I-1 DO IVAL := 8*IVAL + DIGIT[K];
            NEXTCH
         END ↑OCTAL↓ ELSE
         BEGIN IF I > 17 THEN ERROR(2) ELSE
            FOR K := 0 TO I - 1 DO
               BEGIN IF IVAL ≤ MAX10 THEN
                  IVAL := 10*IVAL + DIGIT[K] ELSE
                  BEGIN ERROR(2); IVAL := 0; GOTO 40 END
               END;
40:         SCALE := 0;
            IF CH = Ξ.Ξ THEN
            BEGIN NEXTCH; IF CH = Ξ.Ξ THEN CH := Ξ:Ξ ELSE
               BEGIN RVAL := IVAL; CL := 2;
                  IF ¬(CH IN DIGITS) THEN ERROR(3) ELSE
                  REPEAT R := ORD(CH) - ORD(Ξ0Ξ);
                     RVAL := 10.0*RVAL + R;
                     SCALE := SCALE - 1; NEXTCH
                  UNTIL ¬(CH IN DIGITS);
               END;
            END;
            IF CH = ΞEΞ THEN
            BEGIN IF SCALE = 0 THEN
               BEGIN RVAL := IVAL; CL := 2
               END;
               SIGN := FALSE; NEXTCH;
               IF CH = Ξ+Ξ THEN NEXTCH ELSE
               IF CH = Ξ-Ξ THEN
                  BEGIN NEXTCH; SIGN := TRUE
                  END;
               EXP := 0;
               WHILE CH IN DIGITS DO
               BEGIN EXP := 10*EXP + (ORD(CH) - ORD(Ξ0Ξ));
                  NEXTCH
               END;
               IF SIGN THEN EXP := -EXP;
               SCALE := SCALE + EXP
            END;
            IF SCALE ≠ 0 THEN
            BEGIN R := 1.0;
               IF SCALE < 0 THEN BEGIN FAC := 0.1; SCALE := -SCALE END
                            ELSE FAC := 10.0;
               REPEAT IF ODD(SCALE) THEN R := R*FAC;
                  FAC := SQR(FAC); SCALE := SCALE DIV 2
               UNTIL SCALE = 0; ↑R = 10 ** SCALE↓
               RVAL := RVAL*R
            END;
            IF CL = 2 THEN IVAL := ORD(RVAL)
      END ↑NOT OCTAL↓
```

```
END ↱NUMBER↳ ELSE
BEGIN  ↱ SPECIAL CHARACTER ↳
   IF CH = ΞΞΞΞ THEN  ↱ ALFA OR CHAR CONSTANT ↳
   BEGIN NO := 2; K := 0; BT1 := FALSE;
      REPEAT NEXTCH;
         IF CH = ΞΞΞΞ THEN
         BEGIN NEXTCH; BT1 := CH ≠ ΞΞΞΞ END;
         IF ¬BT1 THEN
            IF K = ALFALENG THEN
            BEGIN ERROR(3); BT1 := TRUE END ELSE
            BEGIN K := K + 1; A[K] := CH END;
      UNTIL BT1;
      IF K =1 THEN  ↱ CHAR CONST ↳
      BEGIN CL:=4; IVAL:= ORD(A[1]) END
      ELSE  ↱ ALFA CONST ↳
      BEGIN IF K ≥ KK THEN KK := K ELSE
         REPEAT A[KK] := Ξ Ξ ; KK := KK - 1 UNTIL KK = K ;
         PACK (A,1,AA);
         CL:=3; IVAL:= ORD(AA);
      END;
   END  ↱ Ξ ↳  ELSE
   BEGIN NO:= SYMNO[CH]; CL:=SYMCL[CH]; IVAL:=0;
       ↱ TEST FOR TWO-CHARACTER SYMBOL ↳
      IF CH = Ξ:Ξ THEN
      BEGIN NEXTCH;
         IF CH=Ξ=Ξ THEN
         BEGIN NO:= 20; NEXTCH END;
      END ELSE
      IF CH =Ξ.Ξ THEN
      BEGIN NEXTCH;
         IF CH = Ξ.Ξ THEN  ↱ : ↳
         BEGIN NO:=19; NEXTCH END;
      END ELSE
      IF CH = Ξ↱Ξ THEN  ↱ SKIP COMMENT ↳
      BEGIN NEXTCH;
         IF CH = Ξ$Ξ THEN
         REPEAT NEXTCH ;
         IF CH ≠ Ξ↳Ξ THEN
         BEGIN CH1 := CH ; NEXTCH ;
            IF CH1 = ΞAΞ THEN ASSCHECK := CH = Ξ+Ξ ELSE
            IF CH1 = ΞCΞ THEN PRCODE := CH = Ξ+Ξ ELSE
            IF CH1 = ΞDΞ THEN DIVCHECK := CH = Ξ+Ξ ELSE
            IF CH1 = ΞOΞ THEN STOFLCHECK := CH = Ξ+Ξ ELSE
            IF CH1 = ΞRΞ THEN ROUNDING := CH = Ξ+Ξ ELSE
            IF CH1 = ΞTΞ THEN
            BEGIN ASSCHECK := CH = Ξ+Ξ ;
               DIVCHECK := CH = Ξ+Ξ ; STOFLCHECK := CH = Ξ+Ξ ;
               ROUNDING := CH = Ξ+Ξ ; INXCHECK := CH = Ξ+Ξ
            END ELSE
            IF CH1 = ΞXΞ THEN INXCHECK := CH = Ξ+Ξ ;
            IF CH ≠ Ξ↳Ξ THEN NEXTCH
         END ;
         UNTIL CH ≠ Ξ,Ξ ;
         WHILE CH ≠ Ξ↳Ξ DO NEXTCH;
         NEXTCH;  GOTO 1 ;
      END ELSE
   NEXTCH;
END  ↱ SPECIAL CHARACTER ↳ ;
END;
END ↱ INSYMBOL ↳ ;

↱ ---------------------------------------------------------- ↳

  PROCEDURE SRCHREC(P:CTP);
      ↱ SEARCHES ONE BLOCK, RETURNS CTPTR ↳
  BEGIN CTPTR := P;
```

```
      WHILE CTPTR ≠ NIL DO
         IF CTPTR↑.NAME = AVAL THEN GOTO 1
         ELSE CTPTR := CTPTR↑.NXTEL;
1:END  ↱ SRCHREC ↓ ;

  PROCEDURE SEARCH;
     ↱ SEARCHES CONTEXTTABLE, RETURNS CTPTR AND DISX = INDEX TO
       DISPLAY ↓
     VAR I : SHRTINT;
  BEGIN FOR I := TOP DOWNTO 0 DO
     BEGIN CTPTR := DISPLAY[I].FNAME;
        WHILE CTPTR ≠ NIL DO
           IF CTPTR↑.NAME = AVAL THEN GOTO 1
           ELSE CTPTR := CTPTR↑.NXTEL;
     END;
1:   DISX := I;
  END  ↱ SEARCH ↓ ;

  PROCEDURE INCONST (VAR V : INTEGER; VAR  P : CTP;  NXT : CTP) ;
     ↱ INPUT PARAMETER : NXT. SEARCHING IS TO BEGIN AT NXT
       OUTPUT PARAMETER : V = VALUE
                          P = TYPE POINTER , P = NIL IF ERROR ↓
     VAR SIGN : BOOLEAN;  PT : CTP;
  BEGIN SIGN := FALSE;  P := NIL;
     IF NO = 7 THEN
     BEGIN SIGN := CL = 2;
        IF CL ≤ 2 THEN
        BEGIN INSYMBOL;  IF NO = 1 THEN ERROR(3)  END;
     END;
     IF NO = 2 THEN
     BEGIN CASE CL OF
1:      P := INTPTR;
2:      P := REALPTR;
3:      P := ALFAPTR;
4:      P := CHARPTR;
        END;
        IF SIGN THEN V := -IVAL ELSE V := +IVAL;
        INSYMBOL;
     END ELSE
     IF NO = 1 THEN
     BEGIN PT := CTPTR;
        SRCHREC(NXT);
        IF CTPTR = NIL THEN SEARCH;
        IF CTPTR = NIL THEN ERROR(12) ELSE
        WITH CTPTR↑ DO
        BEGIN IF KLASS ≠ KONST THEN ERROR(63)
           ELSE
           BEGIN P := CONTYPE;  V := VALUES END;
        END;
        CTPTR := PT;  INSYMBOL;
     END ELSE ERROR(3);
  END ↱ INCONST ↓ ;

↱ ---------------------------------------------------------------- ↓

↱ CODE GENERATING ROUTINES ↓

  PROCEDURE NOOP;
  BEGIN WHILE CP < 4 DO
     BEGIN APPEND(BUF,15,NOP);
        CP := CP + 1;
     END ;
  END  ↱ NOOP ↓  ;

  PROCEDURE GEN15(OP,I,J,K : SHRTINT);
  BEGIN LASTOP := OP ; LASTI := I ;
```

```
      IF CP ≠ 4 THEN
      BEGIN CP := CP + 1; APPEND(BUF,6,OP)  END ELSE
      BEGIN CODE[CA] := BUF;  BUF := OP;  CP := 1;
         CA := CA + 1;  IC := IC + 1;
         IF CA > CODMAX THEN
         BEGIN ERROR(90);  CA := 1;
         END;
      END;
      BUF := ((8*BUF + I)*8 + J)*8 + K ;
END  ↱ GEN15 ↳ ;

  PROCEDURE GEN30(OP,I,J,K : SHRTINT);
  BEGIN LASTOP := OP ; LASTI := I ;
      IF CP = 3 THEN ↱NOOP↳
      BEGIN APPEND(BUF,15,NOP);  CP := 4 END;
      IF CP < 4 THEN
      BEGIN CP := CP + 2;  APPEND(BUF,6,OP)  END ELSE
      BEGIN CODE[CA] := BUF;  BUF := OP;  CP := 2;
         CA := CA + 1;  IC := IC + 1;
         IF CA > CODMAX THEN
         BEGIN ERROR(90);  CA := 1;
         END;
      END;
      BUF := (8*BUF + I)*8 + J ;
      IF K ≥ 0 THEN APPEND(BUF,18,K) ELSE APPEND(BUF,18,777777B+K) ;
END  ↱ GEN30 ↳ ;

  PROCEDURE BXIXJ(I, J : SHRTINT) ;
  ↱TO AVOID B XI XJ INSTRUCTIONS WHENEVER POSSIBLE↳
  BEGIN IF (J = LASTI)∧(LASTOP ≥ 10B)∧(LASTOP ≤ 45B)
     ∧¬(LASTOP IN [20B,21B,43B]) THEN
     BEGIN IF BUF ≥ 0 THEN BUF := BUF - LASTI*64 + I*64 ELSE
        BEGIN BUF := - BUF ;
           BUF := BUF - (7 - LASTI)*64 + (7 - I)*64 ;
           BUF := - BUF
        END ;
        LASTI := I
     END ELSE GEN15(10B,I,J,0)
  END ↱BXIXJ↳ ;

  PROCEDURE INS(FADR : ADDRESS ; FCP, FCA : SHRTINT) ;
  BEGIN IF CA ≠ FCA THEN INSERT(FADR,(4-FCP)*15,CODE[FCA])
                    ELSE INSERT(FADR,(CP-FCP)*15,BUF)
  END ;

↱ ------------------------------------------------------------------ ↳

↱ ROUTINES TO PRINT CODE AND JUMPTABLE AND TO OUTPUT CODE ↳

  PROCEDURE PRTCOMP;
      ↱ GLOBAL ARRAYS : JB1,JB2,JX1,JX2,REL,FS,R1,SH ↳
     VAR II,JJ,KK : CHAR;
        C,CC,S,IT3, OP,I,J,K : SHRTINT;
        M,IT4 : SHRTINT;

     PROCEDURE OUTCH(C : CHAR) ;
     BEGIN WRITE(C) END ;

     PROCEDURE SPACE ;
     BEGIN WRITE(Ξ   Ξ) END ;

  BEGIN
     IC := IC - CA - 1;
     M := -778;  IT4 := -7B;
     FOR IT3 := 1 TO CA DO
     BEGIN BUF := -CODE[IT3];  OUTCH(EOL);
```

```
      OUTCH(Ξ Ξ) ; OUTCH(Ξ Ξ) ; WRITE(IC+IT3 : 6 OCT);  S := 0;
      REPEAT IT := BUF;  APPEND(IT,S+6,M);  OP := -IT;
         C := OP MOD 8;
         IT := BUF;  APPEND(IT,S+9,IT4);  I := -IT;
         IT := BUF;  APPEND(IT,S+12,IT4);  J := -IT;
         IT := BUF;
         IF (OP < 8) ∨ ((OP ≥ 40) ∧ (C ≤ 2)) THEN
         BEGIN S := S+30;  APPEND(IT,S,MASK)  END ELSE
         BEGIN S := S+15;  APPEND(IT,S,IT4)  END;
         K := -IT;

   CC := OP DIV 8;
   II :=CHR(I+ORD(Ξ0Ξ)); JJ := CHR(J+ORD(Ξ0Ξ)); KK :=CHR(K+ORD(Ξ0Ξ));
   OUTCH(EOL); WRITE(BLANK);
   IF CC < 5 THEN
   CASE CC OF
0:    IF OP = 3 THEN
      BEGIN OUTCH(JX1[II]);  OUTCH(JX2[I]);  OUTCH(Ξ Ξ);
         SPACE;  OUTCH(ΞXΞ);  OUTCH(JJ);  OUTCH(Ξ,Ξ);
         WRITE(K : 6 OCT);
      END ELSE
      BEGIN OUTCH(JB1[OP]);  OUTCH(JB2[OP]);  OUTCH(Ξ Ξ);
         SPACE;
         IF OP > 3 THEN
         BEGIN OUTCH(ΞBΞ);  OUTCH(II);  OUTCH(REL[OP]);
            OUTCH(ΞBΞ);  OUTCH(JJ);  OUTCH(Ξ,Ξ);  WRITE(K : 6 OCT);
         END ELSE
         BEGIN IF OP = 2 THEN
            BEGIN OUTCH(ΞBΞ) ; OUTCH(II) ; OUTCH(Ξ+Ξ) END ;
            WRITE(K : 6 OCT)
         END;
      END  ↱ 0 ↳ ;
1:    BEGIN OUTCH(ΞBΞ);  OUTCH(ΞXΞ);  OUTCH(II);  SPACE;
         IF OP > 11 THEN
         BEGIN OUTCH(Ξ¯Ξ);  II := JJ; JJ := KK; KK := II  END;
         OUTCH(ΞXΞ);  OUTCH(JJ);
         C := OP MOD 4;
         IF C ≠ 0 THEN
         BEGIN OUTCH(REL[C]);  OUTCH(ΞXΞ);  OUTCH(KK)  END;
      END  ↱ 1 ↳ ;
2:    BEGIN OUTCH(SH[OP]);  OUTCH(ΞXΞ);  OUTCH(II);  SPACE;
         IF OP < 18 THEN
         BEGIN OUTCH(JJ);  OUTCH(KK)  END ELSE
         BEGIN OUTCH(ΞBΞ);  OUTCH(JJ);  OUTCH(Ξ,Ξ);
            OUTCH(ΞXΞ);  OUTCH(KK);
         END;
      END  ↱ 2 ↳ ;
3:    BEGIN OUTCH(SH[OP DIV 2]);  OUTCH(ΞXΞ);  OUTCH(II);
         SPACE;  OUTCH(ΞXΞ);  OUTCH(JJ);
       IF ODD(OP) THEN OUTCH(Ξ-Ξ) ELSE OUTCH(Ξ+Ξ);
         OUTCH(ΞXΞ);  OUTCH(KK);
      END  ↱ 3 ↳ ;
4:    BEGIN C := OP MOD 4;
         IF OP = 33 THEN
         BEGIN OUTCH(ΞNΞ);  OUTCH(Ξ0Ξ)  END ELSE
         IF C = 3 THEN
         BEGIN IF OP = 35 THEN OUTCH(ΞMΞ) ELSE OUTCH(ΞCΞ);
            OUTCH(ΞXΞ);  OUTCH(II);  SPACE;
            IF OP = 35 THEN OUTCH(JJ) ELSE OUTCH(ΞXΞ) ;
            OUTCH(KK);
         END ELSE
         BEGIN OUTCH(FS[C]);  OUTCH(ΞXΞ);  OUTCH(II);
            SPACE;  OUTCH(ΞXΞ);  OUTCH(JJ);
            IF OP < 36 THEN OUTCH(Ξ*Ξ) ELSE OUTCH(Ξ/Ξ);
            OUTCH(ΞXΞ);  OUTCH(KK);
         END;
```

```
      END  ↱ 4 ↳ ;
   END  ↱ CASE ↳  ELSE
   BEGIN
      OUTCH(ΞSΞ);  OUTCH(FS[CC]);  OUTCH(II);
      SPACE;  OUTCH(R1[C]);  OUTCH(JJ);
      IF C IN [5,7] THEN OUTCH(Ξ-Ξ) ELSE OUTCH(Ξ+Ξ) ;
      IF C ≤ 2 THEN WRITE(K : 6 OCT) ELSE
      BEGIN OUTCH(ΞBΞ);  OUTCH(KK)  END;
   END;

      UNTIL S ≥ 60;
   END;
   IC := IC + CA + 1;  OUTCH(EOL);
END  ↱ PRTCOMP ↳ ;

PROCEDURE PRJMPTAB ;
 ↱ PRINTS OUT JUMPTAB.
  LOADPOINT OF JMPTAB IN PARMLIST.LPJMPTAB ↳
   VAR I : SHRTINT;
BEGIN IF JMPIX > 0 THEN WRITE(EOL,Ξ1Ξ,EOL) ;
   FOR I := 0 TO JMPIX - 1 DO
   WRITE(Ξ  Ξ,PARMLIST.LPJMPTAB+I:6 OCT,Ξ Ξ,JMPTAB[I]:20 OCT,EOL)
END ↱PRJMPTAB↳ ;

PROCEDURE WRITOUT ;
 ↱CALLED AT PROCEDURE END, UPDATING ITS CODE (INSERTING ADDRESSES
  OF CONSTANTS) AND WRITING OUT CODE TOGETHER WITH THE CONSTANTS
  USED IN THIS PROCEDURE↳
   VAR IT2 : SHRTINT;
BEGIN
   NOOP ; CODE[CA] := BUF ;
   FOR IT := 1 TO LCX DO
   WITH CSTTB[IT] DO
   BEGIN
      CA := CA + 1 ;
      IF CA > CODMAX THEN
      BEGIN ERROR(90) ; CA := 1 END ELSE
      BEGIN
         CODE[CA] := VALU ; IT1 := INX ;
         REPEAT
            WITH UNDLAB[IT1] DO
            BEGIN INS(IC,LFTSH,PLACE) ; IT2 := IT1 ; IT1 := SUCC END
         UNTIL IT1 = 0 ;
         UNDLAB[IT2].SUCC := CHNIX ;
         CHNIX := INX ; IC := IC + 1
      END
   END ;
   FOR IT := 1 TO CA DO
   BEGIN PASCLGO↑ := CODE[IT] ; PUT(PASCLGO) END ;
   IF PRCODE THEN
   BEGIN
      IC := IC - LCX ; CA := CA - LCX ; PRTCOMP ;
      FOR IT := 1 TO LCX DO
      BEGIN
         CA := CA+1 ; WRITE(Ξ  Ξ,IC:6 OCT,Ξ Ξ,CODE[CA]:20 OCT,EOL) ;
         IC := IC + 1
      END ;
      IF ¬EOLFLAG THEN
      WRITE((Ξ Ξ):CHCNT+8) ;
   END
END  ↱ WRITOUT ↳  ;

↱ ------------------------------------------------------------ ↳

PROCEDURE MULOPT(VAL:SHRTINT; VAR EXP1,EXP2:SHRTINT; VAR OPT:OPTPWR);
  ↱MULOPT DECIDES IF VAL = 2**EXP1              : OPT = PUREP
```

```
                              = 2**EXP1*(2**EXP2+1) : OPT = POSP
                              = 2**EXP1*(2**EXP2-1) : OPT = NEGP
                                OTHER               : OPT = NOOPT ↓
   VAR E1,E2 : SHRTINT;
BEGIN EXP1 := 0 ; EXP2 := 0 ;
   IF VAL > 0 THEN
   BEGIN E1 := 0;
      WHILE ¬ODD(VAL) DO
      BEGIN VAL := VAL DIV 2;  E1 := E1 + 1  END;
      IF VAL = 1 THEN
      BEGIN OPT := PUREP;  EXP1 := E1  END ELSE
      BEGIN VAL := VAL DIV 2;  E2 := 1;
         IF ODD(VAL) THEN
         BEGIN
            REPEAT VAL := VAL DIV 2;  E2 := E2 + 1;
            UNTIL ¬ODD(VAL);
            IF VAL > 0 THEN OPT := NOOPT ELSE
            BEGIN OPT := NEGP;
               EXP2 := E2;  EXP1 := E1;
            END;
         END ELSE
         BEGIN
            REPEAT VAL := VAL DIV 2;  E2 := E2 + 1;
            UNTIL ODD(VAL);
            IF VAL > 1 THEN OPT := NOOPT  ELSE
            BEGIN OPT := POSP;
               EXP2 := E2;  EXP1 := E1;
            END;
         END;
      END;
   END ELSE OPT := NOOPT;
END ↑MULOPT↓;

FUNCTION LOG2PLUS1(VAL : INTEGER) : INTEGER ;
   VAR E : SHRTINT ;
BEGIN E := 0 ;
   REPEAT VAL := VAL DIV 2 ; E := E + 1
   UNTIL VAL = 0 ;
   LOG2PLUS1 := E
END ↑LOG2PLUS1↓ ;

PROCEDURE SKIP(FNO : SHRTINT) ;
BEGIN
   WHILE (ERRCL[NO] = IRRELSY)∧(FNO ≠ NO) DO
   IF (NO = 38)∧(CL = 2) THEN  ↑RECORD↓
   BEGIN REPEAT INSYMBOL; SKIP(49);
      UNTIL ¬(NO IN [16,26]);
      IF NO = 22 THEN INSYMBOL ;
   END ELSE INSYMBOL;
END  ↑ SKIP ↓  ;

↑ ---------------------------------------------------------------- ↓

PROCEDURE PACKANDNORM(FRP : RG3) ;
BEGIN GEN15(27B,FRP,0,FRP) ; GEN15(24B,FRP,0,FRP) END ;

PROCEDURE JUMPTO(FADR : ADDRESS) ;
BEGIN NOOP ; GEN30(71B,7,0,IC+1) ; GEN30(04B,0,0,FADR) END ;

PROCEDURE LDCST(FVAL : INTEGER) ;
 ↑ ENTERS CONSTANT WITH VALUE FVAL INTO CONSTANTTABLE CSTTB IFF
   NOT YET PRESENT, ELSE CHAINS OCCURRENCES OF FVAL IN CODE
   (THROUGH UNDLAB)   ↓
   VAR IT : SHRTINT;
BEGIN
   RP := RP + 1 ; IF RP = 6 THEN ERROR(33) ;
```

```
GEN30(51B,RP,0,0) ;
FOR IT := 1 TO LCX DO
WITH CSTTB[IT] DO
IF VALU = FVAL THEN
BEGIN
   IF CHNIX = 0 THEN ERROR(75) ELSE
   BEGIN
      WITH UNDLAB[CHNIX] DO
      BEGIN
         IT1 := SUCC ; SUCC := INX ;
         INX := CHNIX ; PLACE := CA ; LFTSH := CP
      END ;
      CHNIX := IT1
   END ;
   GOTO 10
END ;
IF LCX = CSTMAX THEN BEGIN ERROR(84) ; GOTO 10 END ;
IF CHNIX = 0 THEN BEGIN ERROR(75) ; GOTO 10 END ;
LCX := LCX + 1 ;
WITH CSTTB[LCX] DO
BEGIN VALU := FVAL ; INX := CHNIX END ;
WITH UNDLAB[CHNIX] DO
BEGIN IT1 := SUCC ; SUCC := 0 ; PLACE := CA ; LFTSH := CP END ;
CHNIX := IT1 ;
```

```
0:END  ↑ LDCST ↓  ;

  PROCEDURE MULTCODE(FRP : RG3) ;
  ↑GENERATES CODE FOR REAL MULTIPLICATION, ASSUMING FIRST OPERAND
  IN X-RP, THE OTHER IN X-FRP; RESULT IN X-RP↓
  BEGIN IF ROUNDING THEN GEN15(41B,RP,RP,FRP)
                    ELSE GEN15(40B,RP,RP,FRP)
  END ↑MULTCODE↓ ;

  PROCEDURE CHECKBNDS(FRP: RG3; FMIN,FMAX: INTEGER; FADR: ADDRESS) ;
  ↑DONET USE WITH FRP IN [0,7]↓
  BEGIN GEN30(71B,7,0,IC) ;
     IF FMIN ≠ 0 THEN
     BEGIN IF ABS(FMIN) ≥ TWOTO17 THEN LDCST(FMIN) ELSE
        BEGIN RP := RP + 1 ; GEN30(71B,RP,0,FMIN) END ;
        GEN15(37B,0,FRP,RP) ; RP := RP - 1 ;
     END ;
     IF ABS(FMAX) ≥ TWOTO17 THEN LDCST(FMAX) ELSE
     BEGIN RP := RP + 1 ; GEN30(71B,RP,0,FMAX) END ;
     GEN15(37B,RP,RP,FRP) ;
     IF FMIN ≠ 0 THEN GEN15(12B,0,RP,0) ELSE GEN15(12B,0,RP,FRP) ;
     GEN30(03B,3,0,FADR) ; RP := RP - 1 ;
  END ↑CHECKBNDS↓ ;

 PROCEDURE LOADBASE(FRP, FLEVEL : RG3) ;
   VAR I : SHRTINT ;
 BEGIN GEN15(56B,FRP,5,0) ;
   FOR I := 1 TO LEVEL - FLEVEL - 1 DO GEN15(53B,FRP,FRP,0) ;
 END ↑LOADBASE↓ ;

 PROCEDURE TRANSFER(VAR FATTR : ATTR ; FRP : SHRTINT) ;
 ↑DEPENDING ON THE VALUE OF FRP, TRANSFER GENERATES CODE FOR EITHER
  LOADING THE EXPRESSION DESCRIBED BY FATTR INTO REGISTER X-FRP
  OR FOR STORING X-FRP AT LOCATION DESCRIBED BY FATTR (TAKING INTO
  ACCOUNT CDC 6000 HARDWARE FEATURES)↓
     VAR LRP, REG : SHRTINT;  AT : ADDRESS;  BT : BOOLEAN;
 BEGIN WITH FATTR DO
   IF TYPTR ≠ NIL THEN
   CASE KIND OF
   VARBL: BEGIN IF (FRP > 5)^PCKD THEN
            BEGIN PCKD := FALSE ; TRANSFER(FATTR,1) ;
              IT := BITADR + BITSZ ; BT := IT ≠ WORDLENGTH ;
              IF BT THEN GEN15(20B,1,0,IT) ;
              GEN15(43B,0,0,WORDLENGTH-BITSZ) ; GEN15(11B,1,0,1) ;
              GEN15(15B,FRP,FRP,0) ; GEN15(12B,FRP,FRP,1) ;
              IF BT THEN GEN15(20B,FRP,0,WORDLENGTH-IT) ;
              GEN15(54B,FRP,1,0)
            END ELSE
            BEGIN LRP := FRP ;
              IF ACCESS = DRCT THEN
              BEGIN IF FRP < 5 THEN
                BEGIN RP := RP + 1 ; LRP := RP END ELSE
                IF FRP = 5 THEN ERROR(33) ;
                IF BREG IN [0,LEVEL] THEN
                BEGIN IF BREG ≠ 0 THEN REG := 5 ELSE REG := 0 ;
                  GEN30(51B,LRP,REG,DPLMT) ;
                END ELSE
                BEGIN LOADBASE(5,BREG) ; GEN30(52B,LRP,5,DPLMT) END ;
              END ELSE
              IF ACCESS = INXD THEN
              BEGIN IF BREG IN [0,LEVEL] THEN
                BEGIN IF BREG ≠ 0 THEN REG := 5 ELSE REG := 0 ;
                  IF DPLMT = 0 THEN GEN15(53B,LRP,RP,REG) ELSE
                  BEGIN IF REG = 0 THEN GEN30(52B,LRP,RP,DPLMT) ELSE
```

```
                BEGIN GEN30(62B,7,RP,DPLMT) ;
                  GEN15(56B,LRP,7,5)
                END
              END
            END ELSE
            BEGIN GEN30(62B,7,RP,DPLMT) ;
              LOADBASE(5,BREG) ; GEN15(53B,LRP,5,7)
            END
          END ELSE
          BEGIN IF DPLMT = 0 THEN GEN15(53B,LRP,RP,0)
                             ELSE GEN30(52B,LRP,RP,DPLMT)
          END ;
          IF PCKD THEN
          BEGIN IF MASK THEN
             BEGIN IF BITADR + BITSZ ≠ WORDLENGTH THEN
                GEN15(20B,LRP,0,BITADR+BITSZ) ;
                GEN15(43B,0,0,WORDLENGTH-BITSZ) ;
                GEN15(15B,LRP,LRP,0)
             END ELSE
             BEGIN IF BITADR ≠ 0 THEN GEN15(20B,LRP,0,BITADR) ;
                GEN15(21B,LRP,0,WORDLENGTH-BITSZ)
             END ;
             PCKD := FALSE
          END ;
        END
      END ;
SVAL : BEGIN IF FRP < 5 THEN
        BEGIN RP := RP + 1 ; LRP := RP END ELSE
        IF FRP > 5 THEN LRP := FRP ELSE ERROR(33) ;
        IF VAL = 0 THEN GEN15(13B,LRP,0,0)
                   ELSE GEN30(71B,LRP,0,VAL)
      END ;
LVAL : BEGIN IF CTERM ≠ 0 THEN
        BEGIN GEN30(71B,0,0,CTERM) ; GEN15(36B,FRP,RP,0) END ELSE
        IF RP ≠ FRP THEN BXIXJ(FRP,RP)
      END ;
LCOND: BEGIN IF ARITH THEN
        BEGIN IF JMP ≤ 1 THEN
          BEGIN GEN15(43B,FRP,0,0) ; AT := IC + 1 ;
            GEN30(03B,JMP,0,AT) ; GEN30(71B,FRP,0,1) ; NOOP ;
          END ELSE
          BEGIN GEN15(43B,0,0,1) ;
            IF JMP = 2 THEN GEN15(11B,FRP,0,RP)
                       ELSE GEN15(15B,FRP,0,RP) ;
            GEN15(20B,FRP,0,1)
          END
        END ELSE
        BEGIN IF JMP ≠ 0 THEN
          BEGIN GEN15(43B,0,0,59) ; GEN15(17B,FRP,0,RP) END ELSE
          IF RP ≠ FRP THEN BXIXJ(FRP,RP)
        END
      END
  END ↑CASE KIND↓
END ↑TRANSFER↓ ;

PROCEDURE LOADADR(FATTR : ATTR ; FRP : SHRTINT) ;
↑GENERATES CODE FOR LOADING THE ADDRESS OF THE VARIABLE DESCRIBED
 BY FATTR INTO X-FRP↓
  VAR REG : SHRTINT ;
BEGIN WITH FATTR DO
  IF TYPTR ≠ NIL THEN
  IF PCKD THEN ERROR(87) ELSE
  CASE ACCESS OF
  DRCT  : IF BREG IN [0,LEVEL] THEN
          BEGIN IF BREG ≠ 0 THEN REG := 5 ELSE REG := 0 ;
            GEN30(71B,FRP,REG,DPLMT)
```

```
            END ELSE
            BEGIN LOADBASE(5,BREG) ; GEN30(72B,FRP,5,DPLMT) END ;
  INDRCT: IF DPLMT ≠ 0 THEN GEN30(72B,FRP,RP,DPLMT) ELSE
            IF RP ≠ FRP THEN BXIXJ(FRP,RP) ;
  INXD  : IF BREG IN [0,LEVEL] THEN
            BEGIN IF BREG ≠ 0 THEN REG := 5 ELSE REG := 0 ;
              IF DPLMT = 0 THEN GEN15(73B,FRP,RP,REG) ELSE
              BEGIN IF REG = 0 THEN GEN30(72B,FRP,RP,DPLMT) ELSE
                BEGIN GEN30(62B,7,RP,DPLMT) ; GEN15(76B,FRP,7,5) END ;
              END
            END ELSE
            BEGIN GEN30(62B,7,RP,DPLMT) ;
              LOADBASE(5,BREG) ; GEN15(73B,FRP,5,7)
            END ;
  END
END ↱LOADADR↰ ;

 PROCEDURE GENJP(FJPADDR : ADDRESS) ;
  ↱ ACCORDING TO THE GLOBAL ATTRIBUTE RECORD GATTR (DEFINING AN
    EXPRESSION), THIS PROCEDURE PERFORMS CODE GENERATION FOR A
    CONDITIONAL JUMP (ADDRESS = FJPADDR), DEPENDING ON THE TRUTH
    VALUE OF THE (NECESSARILY) BOOLEAN EXPRESSION ↰
 BEGIN
    WITH GATTR DO
    IF TYPTR ≠ BOOLPTR THEN ERROR(57) ELSE
    BEGIN
       IF KIND ≤ SVAL THEN TRANSFER(GATTR,RP) ;
       IF KIND = LCOND THEN
          IF (JMP ≤ 1) ∧ ARITH THEN GEN30(03B,JMP,0,FJPADDR) ELSE
          GEN30(03B,JMP,1,FJPADDR)
       ELSE GEN30(03B,0,RP,FJPADDR)
    END
 END  ↱ GENJP ↰  ;

PROCEDURE ADDRESSVAR(FCTP : CTP ; VAR FATTR : ATTR) ;
↱GENERATES CODE TO ADDRESS THE QUANTITY WITH CTPTR = FCTP AND
 BUILDS UP ITS ATTRIBUTES IN FATTR↰
BEGIN WITH FCTP↑, FATTR DO
  BEGIN KIND := VARBL ;
    IF KLASS = VARS THEN
    BEGIN TYPTR := VTYPE ; PCKD := FALSE ;
      IF VKIND = ACTUAL THEN
      BEGIN ACCESS := DRCT ; DPLMT := VADDR ; BREG := VLEVEL ;
      END ELSE
      BEGIN RP := RP + 1 ; IF RP = 6 THEN ERROR(33) ;
        IF VLEVEL = LEVEL THEN GEN30(51B,RP,5,VADDR) ELSE
        BEGIN LOADBASE(RP,VLEVEL) ; GEN30(52B,RP,RP,VADDR) END ;
        ACCESS := INDRCT ; DPLMT := 0 ;
      END ;
    END ELSE
    IF KLASS = FIELD THEN
    BEGIN TYPTR := FLDTYPE ;
      WITH DISPLAY[DISX] DO
      BEGIN IF OCCUR = CWITH THEN
        BEGIN ACCESS := DRCT ; BREG := CLEV ; DPLMT := FLDADDR + CDSPL
        END ELSE
        BEGIN RP := RP + 1 ; IF RP = 6 THEN ERROR(33) ;
          GEN30(51B,RP,5,VDSPL) ;
          ACCESS := INDRCT ; DPLMT := FLDADDR ;
        END ;
        IF BITWIDTH ≠ 0 THEN
        BEGIN PCKD := TRUE ; BITADR := BITDISPL ; BITSZ := BITWIDTH ;
          MASK := (TYPTR↑.FORM = SYMBOLIC)∨
                  ((TYPTR↑.FORM = NUMERIC)∧(TYPTR↑.MIN ≥ 0)) ;
        END ELSE PCKD := FALSE
      END
```

```
    END ELSE
    BEGIN TYPTR := PROCTYPE ; ACCESS := DRCT ; BREG := PROCLEVEL + 1 ;
      DPLMT := 2 ; PCKD := FALSE ;
      IF PROCKIND = FORMAL THEN ERROR(81) ELSE
      IF LEVEL ≠ BREG THEN ERROR(85)
    END
  END ↑WITH FCTP↑, FATTR↓ ;
END ↑ADDRESSVAR↓ ;


{------------------------------------------------------------------ ↓

PROCEDURE STATEMENT  ; FORWARD ;

PROCEDURE EXPRESSION ; FORWARD ;

PROCEDURE PASSPARAMS ; FORWARD ;

PROCEDURE VARIABLE ;
  VAR LATTR : ATTR ;
BEGIN ADDRESSVAR(CTPTR,LATTR) ;
  INSYMBOL ;
  WHILE NO IN [11,17,18] DO ↑[,.,]↓
  IF NO = 11 THEN ↑[↓
  BEGIN
    REPEAT WITH LATTR DO
      IF TYPTR ≠ NIL THEN
      IF TYPTR↑.FORM ≠ ARRAYS THEN
      BEGIN ERROR(34) ; TYPTR := NIL END ;
      INSYMBOL ; EXPRESSION ;
      IF GATTR.TYPTR ≠ NIL THEN
      BEGIN IF GATTR.TYPTR↑.FORM > SYMBOLIC THEN ERROR(35) ;
        IF LATTR.TYPTR ≠ NIL THEN
        BEGIN PT := LATTR.TYPTR↑.INXTYPE ;
      IF ((GATTR.TYPTR↑.FORM = SYMBOLIC)∨(PT↑.FORM = SYMBOLIC))∧
      (GATTR.TYPTR ≠ PT) THEN ERROR(36) ;
      WITH GATTR DO
      IF KIND = SVAL THEN
      BEGIN IF (LATTR.TYPTR↑.LO > VAL)∨(LATTR.TYPTR↑.HI < VAL)
        THEN ERROR(99) ;
        IT := VAL
      END ELSE
      IF KIND = LVAL THEN IT := CTERM ELSE
      BEGIN TRANSFER(GATTR,RP) ; IT := 0 END ;
      WITH LATTR, TYPTR↑ DO
      BEGIN DPLMT := DPLMT + (IT - LO)*AELTYPE↑.SIZE ;
        IF GATTR.KIND ≠ SVAL THEN
        BEGIN IF INXCHECK THEN CHECKBNDS(RP,LO-IT,HI-IT,INXERR) ;
          IF OPTTYP = NOOPT THEN
          BEGIN GEN30(71B,6,0,AELTYPE↑.SIZE) ;
            GEN15(42B,RP,RP,6)
          END ELSE
          BEGIN IF EXP1 ≠ 0 THEN GEN15(20B,RP,0,EXP1) ;
            IF OPTTYP ≠ PUREP THEN
            BEGIN GEN15(10B,0,RP,0) ; GEN15(20B,RP,0,EXP2) ;
              IF OPTTYP = POSP THEN GEN15(36B,RP,RP,0)
                               ELSE GEN15(37B,RP,RP,0)
            END
          END ;
          IF ACCESS = DRCT THEN ACCESS := INXD ELSE
          BEGIN RP := RP - 1 ; GEN15(36B,RP,RP,RP+1) END
        END
      END ;
    END ↑IF LATTR.TYPTR ≠ NIL↓
  END ↑IF GATTR.TYPTR ≠ NIL↓ ;
  IF LATTR.TYPTR ≠ NIL THEN LATTR.TYPTR := LATTR.TYPTR↑.AELTYPE ;
```

```
    UNTIL NO ≠ 15 ↑,↓ ;
    IF NO ≠ 12 THEN ↑]↓ ERROR(37) ELSE INSYMBOL
  END ↑IF NO = 11↓ ELSE
  IF NO = 17 THEN ↑.↓
  BEGIN INSYMBOL ;
    IF NO = 1 THEN ↑ID↓
    BEGIN IF LATTR.TYPTR ≠ NIL THEN
      BEGIN IF LATTR.TYPTR↑.FORM = RECORDS THEN
        WITH LATTR DO
        BEGIN SRCHREC(TYPTR↑.FSTFLD) ;
          IF CTPTR = NIL THEN
           BEGIN ERROR(39) ; LATTR.TYPTR := NIL END ELSE
          WITH CTPTR↑ DO
          BEGIN TYPTR := FLDTYPE ; DPLMT := DPLMT + FLDADDR ;
            IF BITWIDTH ≠ 0 THEN
            BEGIN BITADR := BITDISPL ; BITSZ := BITWIDTH ;
              MASK := (TYPTR↑.FORM = SYMBOLIC)∨
                      ((TYPTR↑.FORM = NUMERIC)∧(TYPTR↑.MIN ≥ 0)) ;
              PCKD := TRUE
            END ELSE PCKD := FALSE
          END
        END ELSE
        BEGIN ERROR(38) ; LATTR.TYPTR := NIL END ;
      END ;
      INSYMBOL
    END ELSE
    BEGIN ERROR(41) ; LATTR.TYPTR := NIL END
  END ↑IF NO = 17↓ ELSE ↑↑↓
  BEGIN WITH LATTR DO
    IF TYPTR ≠ NIL THEN
    BEGIN TRANSFER(LATTR,RP) ; ACCESS := INDRCT ; DPLMT := 0 ;
      IF TYPTR↑.FORM = FILES THEN TYPTR := TYPTR↑.FELTYPE ELSE
      IF TYPTR↑.FORM = POINTER THEN TYPTR := TYPTR↑.ELTYPE ELSE
      BEGIN ERROR(40) ; TYPTR := NIL END
    END ;
    INSYMBOL
  END ↑↑↓ ;
  GATTR := LATTR
END ↑VARIABLE↓ ;

PROCEDURE FACTOR ;
   VAR LATTR : ATTR ; LPTR : CTP ; LRP, LB6 : SHRTINT ;
       AT : ADDRESS;  LPSVAL, IT2 : INTEGER;

  PROCEDURE ELEMENT ;
  ↑WORKS UP THE NEXT ELEMENT IN THE SET BEEING ANALYSED↓
  BEGIN
    EXPRESSION ;
    IF GATTR.TYPTR ≠ NIL THEN
    IF GATTR.TYPTR↑.FORM > SYMBOLIC THEN ERROR(35) ELSE
    IF (GATTR.TYPTR = REALPTR)∨(GATTR.TYPTR = ALFAPTR) THEN
    ERROR(62) ELSE
    BEGIN IF LATTR.TYPTR ≠ NIL THEN
      IF ((LATTR.TYPTR↑.FORM ≠ NUMERIC)∨(GATTR.TYPTR↑.FORM ≠
      NUMERIC))∧(LATTR.TYPTR ≠ GATTR.TYPTR) THEN ERROR(73) ;
      LATTR.TYPTR := GATTR.TYPTR ;
      IF GATTR.KIND = SVAL THEN INSERT(1,GATTR.VAL,LPSVAL) ELSE
      BEGIN TRANSFER(GATTR,RP) ; GEN15(63B,7,RP,0) ;
        IF LATTR.KIND = SVAL THEN
        BEGIN GEN30(71B,RP,0,1) ; GEN15(22B,RP,7,RP) ;
          LATTR.KIND := LVAL ; LATTR.CTERM := 0
        END ELSE
        BEGIN GEN30(71B,0,0,1) ; GEN15(22B,0,7,0) ;
          RP := RP - 1 ; GEN15(12B,RP,RP,0) ;
        END
      END
```

```
      END ;
    END ↑ELEMENT↓ ;

  BEGIN IF NO = 1 THEN                                        ↑IDENTIFIER↓
    BEGIN SEARCH ;
      IF CTPTR = NIL THEN
      BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
      CASE CTPTR↑.KLASS OF
TYPES: BEGIN ERROR(45) ; GATTR.TYPTR := NIL ; INSYMBOL END ;
KONST: WITH GATTR, CTPTR↑ DO
        BEGIN TYPTR := CONTYPE ;
          IF ABS(VALUES) ≥ TWOTO17 THEN
          BEGIN LDCST(VALUES) ; KIND := LVAL ; CTERM := 0 END ELSE
          BEGIN KIND := SVAL ; VAL := VALUES END;
          INSYMBOL
        END ;
PROC : BEGIN INSYMBOL ;
          IF (CTPTR↑.PROCTYPE = NIL)∨(CTPTR↑.PROCTYPE = CTPTR) THEN
          BEGIN ERROR(46) ; GATTR.TYPTR := NIL END ELSE
          IF NO ≠ 9 THEN ↑(↓
          BEGIN ERROR(79) ; GATTR.TYPTR := NIL END ELSE
          IF CTPTR ≤ PREDEFP THEN                    ↑**INLINE FCT**↓
          BEGIN LPTR := CTPTR ; INSYMBOL ; EXPRESSION ;
            IF GATTR.TYPTR ≠ NIL THEN
            CASE LPTR↑.SEGSIZE OF
    ↑ODD↓ 1: BEGIN IF GATTR.TYPTR↑.FORM ≠ NUMERIC THEN ERROR(47) ;
                TRANSFER(GATTR,RP) ;
                IF GATTR.TYPTR↑.MIN ≥ 0 THEN GEN15(20B,RP,0,59) ELSE
                BEGIN GEN15(10B,0,RP,0) ; GEN15(20B,0,0,59) ;
                  GEN15(13B,RP,RP,0) ;
                END ;
                WITH GATTR DO
                BEGIN TYPTR := BOOLPTR ; KIND := LCOND ; JMP := 2 ;
                  ARITH := TRUE
                END
              END ;
      ↑ORD↓ 2: BEGIN IF GATTR.TYPTR↑.FORM > POWER THEN ERROR(44) ;
                GATTR.TYPTR := INTPTR
              END ;
      ↑CHR↓ 3: BEGIN IF GATTR.TYPTR↑.FORM ≠ NUMERIC THEN ERROR(47) ;
                IF ASSCHECK THEN
                BEGIN TRANSFER(GATTR,RP) ;
                  WITH GATTR DO
                  BEGIN KIND := LVAL ; CTERM := 0 END ;
                  CHECKBNDS(RP,0,63,ASSERR)
                END ;
                GATTR.TYPTR := CHARPTR
              END ;
      ↑EOF↓ 4: BEGIN IF GATTR.TYPTR↑.FORM ≠ FILES THEN ERROR(44) ;
                WITH GATTR DO
                BEGIN DPLMT := DPLMT + 1 ; TRANSFER(GATTR,RP) ;
                  TYPTR := BOOLPTR ; KIND := LCOND ; JMP := 2 ;
                  ARITH := TRUE
                END
              END ;
      ↑ABS↓ 5: BEGIN IF (GATTR.TYPTR ≠ REALPTR)∧(GATTR.TYPTR↑.FORM ≠
                NUMERIC) THEN ERROR(44) ;
                TRANSFER(GATTR,RP) ; GEN15(10B,0,RP,0) ;
                GEN15(21B,0,0,73B) ; GEN15(13B,RP,0,RP) ;
                WITH GATTR DO
                BEGIN IF TYPTR ≠ REALPTR THEN TYPTR := INTPTR ;
                  KIND := LVAL ; CTERM := 0
                END
              END ;
      ↑SQR↓ 6: BEGIN IF (GATTR.TYPTR ≠ REALPTR)∧(GATTR.TYPTR↑.FORM ≠
                NUMERIC) THEN ERROR(44) ;
```

```
            TRANSFER(GATTR,RP) ;
            WITH GATTR DO
            BEGIN IF TYPTR = REALPTR THEN MULTCODE(RP) ELSE
              BEGIN GEN15(42B,RP,RP,RP) ; TYPTR := INTPTR END ;
              KIND := LVAL ; CTERM := 0 ;
            END
          END ;
 (*TRC*) 7: BEGIN IF GATTR.TYPTR ≠ REALPTR THEN ERROR(44) ;
            TRANSFER(GATTR,RP) ;
            GEN15(26B,RP,7,RP) ; GEN15(22B,RP,7,RP) ;
            GEN15(13B,0,0,0) ; GEN15(36B,RP,RP,0) ;
            WITH GATTR DO
            BEGIN TYPTR := INTPTR ; KIND := LVAL ; CTERM := 0 END
          END ;
 (*PRE*) 8,
 (*SUC*) 9: IF (GATTR.TYPTR↑.FORM > SYMBOLIC)∨(GATTR.TYPTR =
          REALPTR)∨(GATTR.TYPTR = ALFAPTR) THEN
          BEGIN ERROR(44) ; GATTR.TYPTR := NIL END ELSE
          BEGIN TRANSFER(GATTR,RP) ;
            WITH GATTR.TYPTR↑ DO
            IF FORM = NUMERIC THEN
            BEGIN IT1 := MIN ; IT2 := MAX END ELSE
            BEGIN IT1 := 0 ; IT2 := FCONST↑.VALUES END ;
            IT := LPTR↑.SEGSIZE ;
            IF (ABS(IT1) ≥ TWOTO17)∨(ABS(IT2) ≥ TWOTO17) THEN
            BEGIN GEN30(71B,0,0,1) ;
              GEN15(47B-IT,RP,RP,0) ;   (*36B OR 37B*)
            END ELSE GEN30(72B,RP,RP,2*IT-17) ;  (*+1 OR -1*)
            IF ASSCHECK∧(GATTR.TYPTR ≠ INTPTR) THEN
            CHECKBNDS(RP,IT1,IT2,ASSERR) ;
            WITH GATTR DO
            BEGIN KIND := LVAL ; CTERM := 0 END
          END
        END (*CASE LPTR↑.SEGSIZE OF*) ;
            END
          END (*CASE LPTR↑.SEGSIZE OF*) ;
          IF NO ≠ 10 THEN (*)*)
          BEGIN ERROR(48) ; GATTR.TYPTR := NIL END ELSE INSYMBOL
        END (*IF CTPTR ≤ PREDEFP*) ELSE                (**EXTERNAL FCT**)
        BEGIN FOR IT := 1 TO RP DO                     (**USERDEF. FCT**)
          BEGIN BXIXJ(6,IT) ; GEN30(51B,6,5,LC+TCT) ;
            TCT := TCT + 1 ; IF TCT > TMAX THEN TMAX := TCT ;
          END ;
          IF CTPTR ≤ EXTPTR THEN  (* SET BIT TO INFORM MONITOR *)
          WITH PARMLIST DO EXTFLAGS := EXTFLAGS∨[CTPTR↑.SEGSIZE] ;
          LRP := RP ; LB6 := B6DPL ; LPTR := CTPTR↑.PROCTYPE ;
          IF LB6 ≠ 3 THEN GEN30(61B,6,6,LB6) ; RP := 0 ;
          PASSPARAMS ;
          IF LB6 ≠ 3 THEN GEN30(61B,6,6,-LB6) ;
          B6DPL := LB6 ;
          FOR IT := LRP DOWNTO 1 DO
          BEGIN TCT := TCT - 1 ; GEN30(51B,IT,5,LC+TCT) END ;
          IF LRP = 5 THEN ERROR(33) ELSE
          BEGIN RP := LRP + 1 ;
            IF LB6 = 3 THEN IT := 2 ELSE IT := LB6 + 2 ;
            GEN30(51B,RP,6,IT) ;
          END ;
          WITH GATTR DO
          BEGIN TYPTR := LPTR ; KIND := LVAL ; CTERM := 0 END
        END
      END ;
VARS ,
FIELD : VARIABLE ;
    END (*CASE CTPTR↑.KLASS OF*) ;
  END (*IF NO = 1*) ELSE
  IF NO = 2 THEN                                       (*CONSTANT*)
  BEGIN WITH GATTR DO
    BEGIN IF ABS(IVAL) ≥ TWOTO17 THEN
```

```
      BEGIN LDCST(IVAL) ; KIND := LVAL ; CTERM := 0 END ELSE
      BEGIN KIND := SVAL ; VAL := IVAL END ;
      CASE CL OF
     1: TYPTR := INTPTR  ;
     2: TYPTR := REALPTR ;
     3: TYPTR := ALFAPTR ;
     4: TYPTR := CHARPTR ;
      END ;
    END ;
    INSYMBOL ;
  END {IF NO = 2} ELSE
  IF NO = 36 THEN                                              {NIL}
  BEGIN WITH GATTR DO
    BEGIN TYPTR := NILPTR ; KIND := SVAL ; VAL := NILVAL END ;
    INSYMBOL
  END ELSE
  IF NO = 5 THEN                                               { ~ }
  BEGIN INSYMBOL ; FACTOR ;
    WITH GATTR DO
    IF TYPTR <> NIL THEN
    IF TYPTR = BOOLPTR THEN
    BEGIN IF KIND = LCOND THEN
      BEGIN IF JMP <= 1 THEN JMP := 1 - JMP ELSE JMP := 5 - JMP ;
      END ELSE
      BEGIN TRANSFER(GATTR,RP) ; GEN15(43B,0,0,59) ;
        GEN15(17B,RP,0,RP) ; KIND := LVAL ; CTERM := 0
      END
    END ELSE ERROR(50)
  END {IF NO = 5} ELSE
  IF NO = 9 THEN                                               { ( }
  BEGIN INSYMBOL ; EXPRESSION ;
    IF NO <> 10 THEN {)}
    BEGIN ERROR(48) ; GATTR.TYPTR := NIL END ELSE INSYMBOL
    END ELSE
    IF NO = 11 THEN                                            { [ }
    BEGIN INSYMBOL ;
      IF NO = 12 THEN {]}
      WITH GATTR DO
      BEGIN TYPTR := LAMPTR ; KIND := SVAL ;
         VAL := 0 ; INSYMBOL
      END ELSE
      BEGIN WITH LATTR DO
        BEGIN TYPTR := NIL ; KIND := SVAL END ; LPSVAL := 0 ;
        ELEMENT ;
        WHILE NO = 15 DO {,}
        BEGIN INSYMBOL ; ELEMENT END ;
        IF LPSVAL <> 0 THEN
        IF LPSVAL >= TWOTO17 THEN
        BEGIN LDCST(LPSVAL) ;
          IF LATTR.KIND = LVAL THEN
          BEGIN RP := RP - 1 ; GEN15(12B,RP,RP,RP+1) END ELSE
          LATTR.KIND := LVAL ; LATTR.CTERM := 0
        END ELSE
        IF LATTR.KIND = SVAL THEN LATTR.VAL := LPSVAL ELSE
        BEGIN GEN30(71B,0,0,LPSVAL) ; GEN15(12B,RP,RP,0) END ;
        IF LATTR.TYPTR <> NIL THEN
        IF LATTR.TYPTR^.FORM = NUMERIC THEN LATTR.TYPTR := PNUMPTR
        ELSE LATTR.TYPTR := LATTR.TYPTR^.PWSET ;
        IF NO <> 12 THEN {]}
        BEGIN ERROR(37) ; LATTR.TYPTR := NIL END ELSE INSYMBOL ;
        GATTR := LATTR
      END ;
    END {IF NO = 11} ELSE
    BEGIN ERROR(42) ; GATTR.TYPTR := NIL END
  END {FACTOR} ;
```

```
  PROCEDURE TERM ;
  VAR LATTR : ATTR ; AT : ADDRESS ; BT1,BT2,BT3,BT4 : BOOLEAN ;
      LEXP1, LEXP2, LMOPCL : SHRTINT ; LOPT : OPTPWR ;
  PROCEDURE CHECKDIV(FRP : RG3) ;
    (*DONET USE WITH FRP = 7*)
    BEGIN IF GATTR.KIND = SVAL THEN
      BEGIN IF GATTR.VAL = 0 THEN ERROR(102) ;
      END ELSE
      IF DIVCHECK THEN
      BEGIN GEN30(71B,7,0,IC) ; GEN30(03B,0,FRP,DIVERR) END ;
    END (*DIVCHECK*) ;
  BEGIN FACTOR ;
    IF NO = 6 THEN (*MULOP*)
    BEGIN LOPT := NOOPT;
      IF (GATTR.KIND = SVAL)^(CL = 1) THEN
      MULOPT(GATTR.VAL,LEXP1,LEXP2,LOPT) ;
      IF LOPT = NOOPT THEN
      BEGIN TRANSFER(GATTR,RP) ;
        WITH LATTR DO
        BEGIN TYPTR := GATTR.TYPTR ; KIND := LVAL ; CTERM := 0 END ;
      END ELSE LATTR := GATTR ;
      REPEAT LMOPCL := CL ; INSYMBOL ; FACTOR ;
        IF (LATTR.TYPTR ≠ NIL)^(GATTR.TYPTR ≠ NIL) THEN
        BEGIN CASE LMOPCL OF
  (* * *) 1: BEGIN BT1 := LATTR.TYPTR↑.FORM = NUMERIC ;
             BT2 := GATTR.TYPTR↑.FORM = NUMERIC ;
             BT3 := LATTR.TYPTR = REALPTR ;
             BT4 := GATTR.TYPTR = REALPTR ;
             IF LOPT ≠ NOOPT THEN
             BEGIN TRANSFER(GATTR,RP) ;
               IF BT4 THEN
               BEGIN TRANSFER(LATTR,RP) ; PACKANDNORM(RP) ;
                 LATTR.TYPTR := REALPTR ;
                 LOPT := NOOPT ; BT3 := TRUE ;
               END ;
               WITH LATTR DO
               BEGIN KIND := LVAL ; CTERM := 0 END ;
             END ELSE
             BEGIN IF (GATTR.KIND = SVAL)^~BT3 THEN
               MULOPT(GATTR.VAL,LEXP1,LEXP2,LOPT) ;
               IF LOPT = NOOPT THEN TRANSFER(GATTR,RP) ;
             END ;
             IF LOPT = NOOPT THEN BEGIN RP1 := RP ; RP := RP - 1 END ;
             IF BT3^BT4 THEN MULTCODE(RP1) ELSE
             IF BT1^BT2 THEN
             BEGIN IF LOPT = NOOPT THEN
               BEGIN GEN15(42B,RP,RP,RP1) ; GEN15(13B,0,0,0) ;
                 GEN15(36B,RP,RP,0)
               END ELSE
               BEGIN IF LEXP1 ≠ 0 THEN GEN15(20B,RP,0,LEXP1) ;
                 IF LOPT ≠ PUREP THEN
                 BEGIN GEN15(10B,0,RP,0) ; GEN15(20B,RP,0,LEXP2) ;
                   IF LOPT = POSP THEN GEN15(36B,RP,RP,0)
                               ELSE GEN15(37B,RP,RP,0) ;
                 END
               END ;
               LATTR.TYPTR := INTPTR ;
             END ELSE
             IF (BT1^BT4)∨(BT2^BT3) THEN
             BEGIN IF BT1 THEN PACKANDNORM(RP) ELSE PACKANDNORM(RP1) ;
               MULTCODE(RP1) ; LATTR.TYPTR := REALPTR
             END ELSE ERROR(50)
           END ;
  (* / *) 2: BEGIN TRANSFER(GATTR,RP) ;
             IF GATTR.TYPTR↑.FORM = NUMERIC THEN
             BEGIN PACKANDNORM(RP) ; GATTR.TYPTR := REALPTR END ;
```

```
          RP := RP - 1 ;
          IF LATTR.TYPTR↑.FORM = NUMERIC THEN
          BEGIN PACKANDNORM(RP) ; LATTR.TYPTR := REALPTR END ;
          IF (GATTR.TYPTR = REALPTR)∧(LATTR.TYPTR = REALPTR) THEN
          BEGIN CHECKDIV(RP+1) ;
            IF ROUNDING THEN GEN15(45B,RP,RP,RP+1)
                        ELSE GEN15(44B,RP,RP,RP+1)
          END ELSE ERROR(50) ;
        END ;
↑ ^ ↓ 3: BEGIN TRANSFER(GATTR,RP) ; RP := RP - 1 ;
          IF (LATTR.TYPTR = GATTR.TYPTR)∧((LATTR.TYPTR↑.FORM =
          POWER)∨(LATTR.TYPTR = BOOLPTR)) THEN GEN15(11B,RP,RP,RP+1)
          ELSE ERROR(50)
        END ;
↑DIV↓ 4: BEGIN IF GATTR.KIND = SVAL THEN
            MULOPT(GATTR.VAL,LEXP1,LEXP2,LOPT) ;
          IF (LATTR.TYPTR↑.FORM = NUMERIC)∧(GATTR.TYPTR↑.FORM =
          NUMERIC) THEN
          BEGIN IF LOPT = PUREP THEN
            BEGIN IF LEXP1 ≠ 0 THEN GEN15(21B,RP,0,LEXP1) ;
              IF LATTR.TYPTR↑.MIN < 0 THEN
              BEGIN GEN15(13B,0,0,0) ; GEN15(36B,RP,RP,0) END
            END ELSE
            BEGIN TRANSFER(GATTR,RP) ; CHECKDIV(RP) ;
              PACKANDNORM(RP) ; RP := RP - 1 ;
              GEN15(27B,RP,0,RP) ;
              GEN15(44B,RP,RP,RP+1) ; GEN15(26B,RP,7,RP) ;
              GEN15(22B,RP,7,RP) ; GEN15(13B,0,0,0) ;
             GEN15(36B,RP,RP,0) ;
            END ;
            LATTR.TYPTR := INTPTR
          END ELSE ERROR(50)
        END ;
↑MOD↓ 5: BEGIN IF GATTR.KIND = SVAL THEN
            MULOPT(GATTR.VAL,LEXP1,LEXP2,LOPT) ;
          IF (LATTR.TYPTR↑.FORM = NUMERIC)∧(GATTR.TYPTR↑.FORM =
          NUMERIC) THEN
          BEGIN IF LOPT = PUREP THEN
            BEGIN GEN15(10B,0,RP,0) ; GEN15(21B,0,0,LEXP1) ;
              GEN15(20B,0,0,LEXP1) ; GEN15(37B,RP,RP,0) ;
            END ELSE
            BEGIN TRANSFER(GATTR,RP) ;
              CHECKDIV(RP) ; GEN15(27B,6,0,RP) ; GEN15(24B,6,0,6) ;
              RP1 := RP ; RP := RP - 1 ;
              GEN15(27B,0,0,RP) ; GEN15(44B,6,0,6) ;
              GEN15(26B,6,7,6) ; GEN15(22B,6,7,6) ;
              GEN15(42B,6,RP1,6) ; GEN15(37B,RP,RP,6) ;
            END ;
            LATTR.TYPTR := INTPTR
          END ELSE ERROR(50)
        END ;
        END ↑CASE LMOPCL OF↓
      END ;
      LOPT := NOOPT
    UNTIL NO ≠ 6 ; ↑MULOP↓
    GATTR := LATTR
  END ↑IF NO = 6↓
END ↑TERM↓ ;

PROCEDURE SIMPLEEXP ;
VAR LATTR : ATTR ; LADOPCL : SHRTINT ; LFG,BT1,BT2 : BOOLEAN ;
BEGIN LFG := FALSE ;
  IF NO = 7 THEN ↑ADDOP↓
  BEGIN IF CL = 2 THEN LFG := TRUE ELSE IF CL = 3 THEN ERROR(51) ;
    INSYMBOL ;
  END ;
```

```
TERM ;
IF LFG∨(NO = 7) THEN
BEGIN WITH LATTR DO
  BEGIN TYPTR := GATTR.TYPTR ; KIND := LVAL ; CTERM := 0 ;
    IF TYPTR ≠ NIL THEN
    BEGIN TRANSFER(GATTR,RP) ;
      IF LFG THEN
      BEGIN GEN15(13B,0,0,0) ;
        IF (TYPTR↑.FORM = NUMERIC)∨(TYPTR = REALPTR) THEN
        GEN15(37B,RP,0,RP) ELSE ERROR(50)
      END
    END
  END ;
  WHILE NO = 7 DO
  BEGIN LADOPCL := CL ; INSYMBOL ; TERM ;
    IF (LATTR.TYPTR ≠ NIL)∧(GATTR.TYPTR ≠ NIL) THEN
    BEGIN BT1 := LATTR.TYPTR↑.FORM = NUMERIC ;
      BT2 := GATTR.TYPTR↑.FORM = NUMERIC ;
      IF BT1∧BT2∧(GATTR.KIND = SVAL) THEN
      BEGIN WITH LATTR DO
        BEGIN TYPTR := INTPTR ;
          CASE LADOPCL OF
          1: CTERM := CTERM + GATTR.VAL ;
          2: CTERM := CTERM - GATTR.VAL ;
          3: ERROR(50)
          END
        END ELSE
        BEGIN TRANSFER(GATTR,RP) ; RP1 := RP ; RP := RP - 1 ;
          IF BT1∧BT2 THEN
          BEGIN LATTR.TYPTR := INTPTR ;
            CASE LADOPCL OF
            1: GEN15(36B,RP,RP,RP1) ;
            2: GEN15(37B,RP,RP,RP1) ;
            3: ERROR(50)
            END
          END ELSE
          BEGIN IF BT1 THEN
            BEGIN PACKANDNORM(RP) ; LATTR.TYPTR := REALPTR
            END ELSE
            IF BT2 THEN
            BEGIN PACKANDNORM(RP1) ; GATTR.TYPTR := REALPTR
            END ;
            IF (LATTR.TYPTR = REALPTR)∧(GATTR.TYPTR = REALPTR) THEN
            BEGIN IF LADOPCL ≤ 2 THEN
              BEGIN IF ROUNDING THEN GEN15(33B+LADOPCL,RP,RP,RP1)
                                ELSE GEN15(27B+LADOPCL,RP,RP,RP1) ;
                GEN15(24B,RP,0,RP)
              END ELSE ERROR(50)
            END ELSE
            IF LATTR.TYPTR = GATTR.TYPTR THEN
            BEGIN IF (LATTR.TYPTR = BOOLPTR)∧(LADOPCL = 3) THEN
              GEN15(12B,RP,RP,RP1) ELSE
              IF LATTR.TYPTR↑.FORM = POWER THEN
              CASE LADOPCL OF
              1: GEN15(13B,RP,RP,RP1) ; ↗NON-STANDARD PASCAL↘
              2: GEN15(15B,RP,RP,RP1) ;
              3: GEN15(12B,RP,RP,RP1) ;
              END ELSE ERROR(50)
            END ELSE ERROR(50)
          END
        END
      END
    END ↗WHILE NO = 7↘ ;
    GATTR := LATTR
END ↗IF LFG∨(NO = 7)↘
```

```
  END ↑SIMPLEEXP↓ ;

  PROCEDURE EXPRESSION ;
  VAR LATTR : ATTR ; LRELOPCL, LCST : SHRTINT ;
    LOF,BT1,BT2 : BOOLEAN ; AT : ADDRESS ;
  BEGIN SIMPLEEXP ;
    IF NO = 8 THEN ↑RELOP↓
    BEGIN LOF := TRUE ;
      WITH GATTR DO
      IF (KIND = SVAL)∧((CL = 7)∨(VAL = 0)∧(TYPTR ≠ BOOLPTR)∧
      ¬(CL IN [1,3])) THEN
      BEGIN LOF := FALSE ; LCST := VAL END ELSE
      BEGIN IF TYPTR ≠ NIL THEN
        IF TYPTR↑.FORM IN [ARRAYS,RECORDS] THEN
        BEGIN IF ACCESS = DRCT THEN
          BEGIN RP := RP + 1 ; IF RP = 6 THEN ERROR(33) END ;
          DPLMT := DPLMT - 1 ; LOADADR(GATTR,RP) ;
        END ELSE TRANSFER(GATTR,RP)
      END ;
      WITH LATTR DO
      BEGIN TYPTR := GATTR.TYPTR ; KIND := LCOND END ;
      LRELOPCL := CL ; INSYMBOL ; SIMPLEEXP ;
      IF (LATTR.TYPTR ≠ NIL)∧(GATTR.TYPTR ≠ NIL) THEN
      BEGIN WITH GATTR DO
        IF (KIND = SVAL)∧(VAL = 0)∧(TYPTR ≠ BOOLPTR)∧
        ¬(LRELOPCL IN [2,4]) ∧ LOF THEN LOF := FALSE ELSE
        BEGIN IF TYPTR↑.FORM IN [RECORDS,ARRAYS] THEN
          BEGIN DPLMT := DPLMT - 1 ; LOADADR(GATTR,6) ;
            IF ACCESS = DRCT THEN RP := RP + 1 ;
          END ELSE TRANSFER(GATTR,RP)
        END ;
        IF LOF THEN
        BEGIN RP1 := RP ; RP := RP - 1 END ;
        IF LRELOPCL = 7 THEN ↑IN↓
        BEGIN IF (GATTR.TYPTR↑.FORM = POWER)∧(GATTR.TYPTR↑.ELSET =
          LATTR.TYPTR)∨(GATTR.TYPTR = PNUMPTR)∧(LATTR.TYPTR↑.FORM =
          NUMERIC) THEN
          BEGIN IF LOF THEN
            BEGIN GEN15(63B,7,RP,0) ; GEN15(23B,RP,7,RP1) ;
              GEN15(20B,RP,0,59) ;
            END ELSE GEN15(20B,RP,0,59-LCST) ;
            WITH LATTR DO
            BEGIN JMP := 2 ; ARITH := TRUE END ;
          END ELSE ERROR(50)
        END ↑IF LRELOPCL = 7↓ ELSE
        BEGIN IF (LATTR.TYPTR = BOOLPTR)∧(GATTR.TYPTR = BOOLPTR) THEN
          BEGIN CASE LRELOPCL OF
           1: BEGIN GEN15(15B,RP,RP1,RP) ; LATTR.JMP := 0 END ;
           2: BEGIN GEN15(15B,RP,RP,RP1) ; LATTR.JMP := 1 END ;
           3: BEGIN GEN15(15B,RP,RP1,RP) ; LATTR.JMP := 1 END ;
           4: BEGIN GEN15(15B,RP,RP,RP1) ; LATTR.JMP := 0 END ;
           5: BEGIN GEN15(13B,RP,RP,RP1) ; LATTR.JMP := 0 END ;
           6: BEGIN GEN15(13B,RP,RP,RP1) ; LATTR.JMP := 1 END
            END ↑CASE↓ ;
            LATTR.ARITH := FALSE
          END ↑IF↓ ELSE
          BEGIN BT1 := LATTR.TYPTR↑.FORM = NUMERIC ;
            BT2 := GATTR.TYPTR↑.FORM = NUMERIC ;
            IF BT1∧(GATTR.TYPTR = REALPTR) THEN
            BEGIN IF LOF THEN PACKANDNORM(RP) ;
              LATTR.TYPTR := REALPTR
            END ;
            IF BT2∧(LATTR.TYPTR = REALPTR) THEN
            BEGIN IF LOF THEN PACKANDNORM(RP1) ;
              GATTR.TYPTR := REALPTR
            END ;
```

```
 IF BT1^BT2v(LATTR.TYPTR = GATTR.TYPTR)^(LATTR.TYPTR↑.FORM
 ≤ POINTER)v(LATTR.TYPTR↑.FORM = POINTER)^(GATTR.TYPTR =
 NILPTR)v(GATTR.TYPTR↑.FORM = POINTER)^(LATTR.TYPTR =
 NILPTR) THEN
 BEGIN IF LATTR.TYPTR = REALPTR THEN IT := 31B ELSE IT := 37B ;
    CASE LRELOPCL OF
1:  BEGIN IF LOF THEN GEN15(IT,RP,RP,RP1) ; LATTR.JMP := 2
    END ;
2:  BEGIN IF LOF THEN GEN15(IT,RP,RP1,RP) ; LATTR.JMP := 3
    END ;
3:  BEGIN IF LOF THEN GEN15(IT,RP,RP,RP1) ; LATTR.JMP := 3
    END ;
4:  BEGIN IF LOF THEN GEN15(IT,RP,RP1,RP) ; LATTR.JMP := 2
    END ;
5:  BEGIN IF LOF THEN GEN15(37B,0,RP,RP1)
                 ELSE BXIXJ(0,RP) ;
      LATTR.JMP := 0
    END ;
6:  BEGIN IF LOF THEN GEN15(37B,0,RP,RP1)
                 ELSE BXIXJ(0,RP) ;
      LATTR.JMP := 1
    END ;
    END ↑CASE↓ ;
    LATTR.ARITH := TRUE
 END ↑IF BT1^BT2v(...↓ ELSE
 IF (LATTR.TYPTR↑.FORM = POWER)^(GATTR.TYPTR↑.FORM =
 POWER) THEN
 BEGIN IF (LATTR.TYPTR = GATTR.TYPTR)v(LATTR.TYPTR = LAMPTR)
   v(GATTR.TYPTR = LAMPTR) THEN
   BEGIN CASE LRELOPCL OF
 1, 4: ERROR(83) ;
    2: BEGIN IF LOF THEN GEN15(15B,0,RP,RP1) ELSE ERROR(89) ;
         LATTR.JMP := 1
       END ;
    3: BEGIN IF LOF THEN GEN15(15B,0,RP1,RP) ELSE ERROR(89) ;
         LATTR.JMP := 1
       END ;
 5, 6: BEGIN IF LOF THEN GEN15(13B,0,RP,RP1)
                    ELSE BXIXJ(0,RP) ;
         LATTR.JMP := LRELOPCL - 5
       END ;
     END ↑CASE↓ ;
     LATTR.ARITH := TRUE
   END ELSE ERROR(50)
 END ↑IF (LATTR...↓ ELSE
 IF (LATTR.TYPTR = GATTR.TYPTR)^((LATTR.TYPTR↑.FORM =
 RECORDS)v(LATTR.TYPTR↑.FORM = ARRAYS)) THEN
 BEGIN IF LRELOPCL ≤ 4 THEN ERROR(94) ELSE
   BEGIN IF RP ≥ 4 THEN ERROR(33) ;
     GEN30(61B,7,0,GATTR.TYPTR↑.SIZE) ; NOOP ; AT := IC ;
     GEN15(53B,RP1,6,7) ; GEN15(53B,RP1+1,RP,7) ;
     GEN15(37B,0,RP1+1,RP1) ; GEN30(03B,1,0,IC+2) ;
     GEN30(61B,7,7,-1) ; GEN30(05B,7,0,AT) ; NOOP ;
     WITH LATTR DO
     BEGIN KIND := LCOND ; ARITH := TRUE ;
       JMP := LRELOPCL - 5
     END ;
   END
 END ELSE ERROR(50)
 END ↑IF ≠ BOOLPTRS↓ ;
 END ↑IF LRELOPCL ≤ 6↓
 END ↑IF TYPTRS ≠ NIL↓ ;
 LATTR.TYPTR := BOOLPTR ;
 GATTR := LATTR
 END ↑IF NO = 8↓
END ↑EXPRESSION↓ ;
```

```
(* ------------------------------------------------------------------ *)

PROCEDURE PASSPARAMS ;
(*PERFORMS CODE GENERATION FOR PASSING PARAMETERS TO CALLED
  PROCEDURE AND JUMPING TO ITS ENTRY*)
VAR LPL,LPC,LOSP,IT2 : SHRTINT ; LPA,AT : ADDRESS; LPK,BT : BOOLEAN;
    LFP : CTP ;
BEGIN WITH CTPTR↑ DO
  BEGIN LPA := PROCADDR ; LPL := PROCLEVEL ;
    LFP := FORMALS ; LPK := PROCKIND = ACTUAL
  END ;
  LOSP := 3 ; LPC := 0 ;
  IF NO = 9 THEN (*(*
  BEGIN REPEAT
    BEGIN IF LFP = NIL THEN
      BEGIN IF LPK THEN
        BEGIN ERROR(72) ; SKIP(49) ; GOTO 1 END ;
        BT := FALSE
      END ELSE
      BT := LFP↑.KLASS = PROC ;
      INSYMBOL ;
      IF BT THEN (*PROC/FCT TO BE PASSED*)
      BEGIN IF NO ≠ 1 THEN (*ID*) ERROR(11) ELSE
        BEGIN SEARCH ;
          IF CTPTR = NIL THEN
          BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
          IF CTPTR↑.KLASS ≠ PROC THEN ERROR(60) ELSE
          BEGIN WITH CTPTR↑ DO
            BEGIN PT := LFP↑.PROCTYPE ;
              IF PROCTYPE ≠ CTPTR THEN (*PARAM. IS FCT*)
              BEGIN IF (PT = LFP)∨(PROCTYPE ≠ PT)∧((PROCTYPE↑.FORM
                ≠ NUMERIC)∨(PT↑.FORM ≠ NUMERIC)) THEN ERROR(60)
              END ELSE (*PARAM. IS PROC*)
              IF PT ≠ LFP THEN ERROR(60) ;
              IF CTPTR ≤ PREDEFP THEN
              BEGIN ERROR(83) ; SKIP(49) ; GOTO 1 END ELSE
              IF CTPTR ≤ EXTPTR THEN  (* SET BIT TO INFORM MONITOR *)
              WITH PARMLIST DO EXTFLAGS := EXTFLAGS∨[CTPTR↑.SEGSIZE] ;
              IF PROCTYPE ≠ NIL THEN
              IF PROCKIND = ACTUAL THEN
              BEGIN PT := FORMALS ; IT1 := 0 ; IT2 := 0 ;
                WHILE PT ≠ NIL DO
                WITH PT↑ DO
                BEGIN IF VTYPE ≠ NIL THEN
                  IF (KLASS = VARS)∧((VKIND = FORMAL)∨
                  (VKIND = ACTUAL)∧(VTYPE↑.SIZE ≠ 1)) THEN
                  APPEND(IT1,1,1) ELSE APPEND(IT1,1,0);
                  IT2 := IT2 + 1; PT := NXTEL
                END;
                APPEND(IT1,17-IT2,0) ; IF IT2 > 17 THEN ERROR(103) ;
                GEN30(71B,6,0,PROCADDR) ;
                IF IT1 ≠ 0 THEN
                BEGIN GEN30(71B,0,0,IT1) ; GEN15(20B,0,0,43) ;
                  GEN15(12B,6,6,0) ;
                END ;
                IF PROCLEVEL ≠ 0 THEN
                BEGIN IF PROCLEVEL = LEVEL THEN GEN15(76B,1,5,0)
                  ELSE LOADBASE(1,PROCLEVEL) ;
                  GEN15(20B,1,0,18) ; GEN15(12B,6,6,1) ;
                END
              END ELSE
              BEGIN IF PROCLEVEL = LEVEL THEN
                GEN30(51B,1,5,PROCADDR) ELSE
                BEGIN LOADBASE(1,PROCLEVEL) ; GEN30(52B,1,1,PROCADDR)
                END ;
```

```
          GEN15(10B,6,1,0)
        END ;
      END {WITH CTPTR↑}
    END {IF CTPTR↑.KLASS = PROC} ;
    INSYMBOL
  END {NO = 1}
END ELSE {EXPRESSION TO BE PASSED}
BEGIN
B6DPL := LDSP ; EXPRESSION ;
IF GATTR.TYPTR ≠ NIL THEN
BEGIN IF LPK THEN
  BEGIN IF LFP↑.VTYPE ≠ NIL THEN
    IF (LFP↑.VKIND = ACTUAL)∧(LFP↑.VTYPE↑.SIZE = 1) THEN
    BEGIN IF GATTR.KIND = VARBL THEN
      BEGIN TRANSFER(GATTR,RP) ; GEN15(10B,6,RP,0) END
      ELSE TRANSFER(GATTR,6) ;
      IF (GATTR.TYPTR↑.FORM = NUMERIC)∧(LFP↑.VTYPE = REALPTR) THEN
      BEGIN PACKANDNORM(6) ; GATTR.TYPTR := REALPTR END ;
    END ELSE
    BEGIN IF GATTR.KIND = VARBL THEN LOADADR(GATTR,6)
                                ELSE ERROR(86) ;
    END ;
    PT := LFP↑.VTYPE;
    IF PT ≠ NIL THEN
    IF GATTR.TYPTR ≠ PT THEN
    IF (GATTR.TYPTR↑.FORM ≠ NUMERIC)∨(PT↑.FORM ≠ NUMERIC) THEN
    IF (GATTR.TYPTR ≠ NILPTR)∨(PT↑.FORM ≠ POINTER) THEN
    IF (GATTR.TYPTR ≠ LAMPTR)∨(PT↑.FORM ≠ POWER) THEN
    ERROR(60) ;
  END {IF LPK} ELSE
  BEGIN IF GATTR.KIND ≠ VARBL THEN TRANSFER(GATTR,6) ELSE
    BEGIN LOADADR(GATTR,1) ;
      IF LPL = LEVEL THEN GEN30(51B,2,5,LPA) ELSE
      BEGIN LOADBASE(2,LPL) ; GEN30(52B,2,2,LPA) END ;
      IF LPC ≠ 0 THEN GEN15(20B,2,0,LPC) ;
      IF CP = 1 THEN AT := IC ELSE AT := IC + 1 ;
      GEN30(03B,3,2,AT) ; GEN15(53B,1,1,0) ; NOOP ;
      GEN15(10B,6,1,0)
    END
  END ;
  END {IF GATTR.TYPTR ≠ NIL} ;
  END {EXP} ;
  GEN30(51B,6,6,LDSP) ;
  LDSP := LDSP + 1 ; LPC := LPC + 1 ; RP := 0 ;
  IF LPK THEN LFP := LFP↑.NXTEL
END
UNTIL NO ≠ 15 ; {,}
IF NO ≠ 10 THEN {)} BEGIN ERROR(48) ; SKIP(49) END ELSE INSYMBOL
END {IF NO = 9} ;
IF LFP ≠ NIL THEN ERROR(72) ;
IF LPK THEN
BEGIN IF LPL ≠ 0 THEN
  BEGIN IF LPL = LEVEL THEN GEN15(76B,6,5,0) ELSE
    BEGIN LOADBASE(1,LPL) ; GEN15(10B,6,1,0) END
  END ;
  GEN30(71B,7,0,IC+1) ; GEN30(04B,0,0,LPA) ;
END ELSE
BEGIN IF LPL = LEVEL THEN GEN30(51B,1,5,LPA) ELSE
  BEGIN LOADBASE(1,LPL) ; GEN30(52B,1,1,LPA) END ;
  GEN15(63B,7,1,0) ; GEN15(20B,1,0,42) ;
  GEN15(73B,6,1,0) ; GEN30(71B,7,0,IC+1) ;
  GEN30(02B,7,0,0) ;
END ;
NOOP ;
```

```
*  BOUPL := 3
 END ↑PASSPARAMS↓ ;

 PROCEDURE ASSIGN ;
 VAR LATTR : ATTR ; AT : ADDRESS;
 BEGIN VARIABLE ; LATTR := GATTR ;
   IF NO ≠ 20 THEN ↑:=↓
   BEGIN IF GATTR.TYPTR ≠ NIL THEN ERROR(52) ; SKIP(20) ;
     IF NO ≠ 20 THEN
     BEGIN IF GATTR.TYPTR = NIL THEN ERROR(52) ; GOTO 1 END
   END ;
   INSYMBOL ; EXPRESSION ;
   IF GATTR.TYPTR = NIL THEN SKIP(49) ELSE
   IF LATTR.TYPTR ≠ NIL THEN
   BEGIN WITH GATTR DO
     IF TYPTR↑.FORM ≥ ARRAYS THEN
     BEGIN DPLMT := DPLMT - 1 ; LOADADR(GATTR,2) ; RP := 2 END ELSE
     IF (KIND = LVAL)∧(CTERM ≠ 0) THEN
     WITH LATTR.TYPTR↑ DO
     IF (FORM = NUMERIC)∧(ABS(MIN) < TWOTO17)∧(ABS(MAX) < TWOTO17) THEN
     GEN30(72B,6,RP,CTERM) ELSE TRANSFER(GATTR,6) ELSE
     IF KIND = VARBL THEN
     BEGIN TRANSFER(GATTR,RP) ; GEN15(10B,6,RP,0) END ELSE
     TRANSFER(GATTR,6) ;
     IF GATTR.KIND ≠ SVAL THEN RP := RP - 1 ;
     IF (LATTR.TYPTR = REALPTR)∧(GATTR.TYPTR↑.FORM = NUMERIC) THEN
     PACKANDNORM(6) ELSE
     IF LATTR.TYPTR ≠ GATTR.TYPTR THEN
     IF (LATTR.TYPTR↑.FORM ≠ NUMERIC)∨(GATTR.TYPTR↑.FORM ≠ NUMERIC)
     THEN
     IF (LATTR.TYPTR↑.FORM ≠ POINTER)∨(GATTR.TYPTR ≠ NILPTR) THEN
     IF (LATTR.TYPTR↑.FORM ≠ POWER)∨(GATTR.TYPTR ≠ LAMPTR) THEN
     ERROR(53) ;
     WITH LATTR.TYPTR↑ DO
     IF FORM ≤ POWER THEN
     BEGIN IF (FORM = NUMERIC)∧(LATTR.TYPTR ≠ INTPTR) THEN
       BEGIN WITH GATTR DO
         IF KIND = SVAL THEN
         BEGIN IF (MIN > VAL)∨(MAX < VAL) THEN ERROR(101) ;
         END ELSE
         IF ASSCHECK THEN CHECKBNDS(6,MIN,MAX,ASSERR)
       END ELSE
       IF (FORM = SYMBOLIC)∧(LATTR.TYPTR ≠ REALPTR)
       ∧(LATTR.TYPTR ≠ ALFAPTR)∧(GATTR.KIND ≠ SVAL)∧ASSCHECK THEN
       CHECKBNDS(6,0,FCONST↑.VALUES,ASSERR) ;
       TRANSFER(LATTR,6)
     END ELSE
     IF FORM ≥ CLASSS THEN ERROR(53) ELSE
     BEGIN LATTR.DPLMT := LATTR.DPLMT - 1 ; LOADADR(LATTR,1) ;
       GEN30(61B,7,0,SIZE) ; NOOP ; AT := IC ;
       GEN15(53B,3,2,7) ; GEN15(10B,6,3,0) ; GEN15(53B,6,1,7) ;
       GEN30(61B,7,7,-1) ; GEN30(05B,0,7,AT)
     END
   END ↑IF LATTR.TYPTR = NIL↓ ;
```

```
*END ↱ASSIGN↳ ;

   PROCEDURE VARIAB ;
   BEGIN IF NO ≠ 1 THEN
     BEGIN ERROR(49) ; GATTR.TYPTR := NIL END ELSE
     BEGIN SEARCH ;
       IF CTPTR = NIL THEN
       BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
       IF CTPTR↑.KLASS ≤ PROC THEN
       BEGIN ERROR(32) ; INSYMBOL ; GATTR.TYPTR := NIL END
       ELSE VARIABLE
     END
   END ↱VARIAB↳ ;

     PROCEDURE GETPUT(LPSW : SHRTINT) ;
        CONST EOR = 107B, REST = 106B, RWRT = 122B ;

     BEGIN
        IF NO ≠ 9 THEN  ↱ SY ≠ ( ↳
        BEGIN ERROR(79) ; SKIP(49) ; GOTO 10 END ;
        INSYMBOL ; VARIAB ;
        WITH GATTR DO
        IF TYPTR ≠ NIL THEN
        BEGIN
           IF TYPTR↑.FORM = FILES THEN
           BEGIN
              DPLMT := DPLMT + 1 ;
              IF ACCESS = DRCT THEN
              BEGIN IF BREG IN [0,LEVEL] THEN
                BEGIN IF BREG ≠ 0 THEN BREG := 5;
                  GEN30(61B,7,BREG,DPLMT);
                END ELSE
                BEGIN LOADBASE(5,BREG); GEN30(62B,7,5,DPLMT) END;
           END ELSE GEN30(62B,7,RP,DPLMT);
           CASE LPSW OF
     ↱EOR↳       1: JUMPTO(EOR) ;
     ↱GET,RST↳ 2,4: BEGIN IF LPSW = 4 THEN
                     BEGIN GEN15(76B,7,7,0);
                      GEN30(51B,7,5,LC+TCT);  TCT := TCT+1;
                      IF TCT > TMAX THEN TMAX := TCT;
                      JUMPTO(REST);  TCT := TCT-1;
                      GEN30(51B,1,5,LC+TCT); GEN15(63B,7,1,0);
                     END;
                     IF TYPTR↑.FELTYPE=CHARPTR THEN JUMPTO(GETC)
                                               ELSE JUMPTO(GETB);
                    END;
     ↱PUT↳       3: IF TYPTR↑.FELTYPE = CHARPTR THEN JUMPTO(PUTC)
                                               ELSE JUMPTO(PUTB) ;
     ↱RWT↳       5: JUMPTO(RWRT)
           END ;
        END ELSE
        IF (TYPTR↑.FORM = POINTER)^(LPSW=4) THEN
        BEGIN
           TRANSFER(GATTR,RP) ; GEN15(10B,6,RP,0) ;
           RP := RP - 1 ; ADDRESSVAR(TYPTR↑.DOMAIN,GATTR) ;
           TRANSFER(GATTR,6)
        END ELSE ERROR(44) ;
   END ;
   IF NO ≠ 10 THEN  ↱ SY ≠ ) ↳
   BEGIN ERROR(48) ; SKIP(49) ; END ELSE INSYMBOL ;
```

```
  END  (* GETPUT *)  ;

  PROCEDURE ALLC ;
    VAR BT1 : BOOLEAN ; LATTR : ATTR ;
  BEGIN
    IF NO # 9 THEN  (* SY # ( *)
    BEGIN ERROR(79) ; SKIP(49) ; GOTO 10 END ;
    INSYMBOL;  VARIAB;  PT := NIL;
    WITH GATTR DO
    IF TYPTR # NIL THEN
    WITH TYPTR^ DO
    IF FORM # POINTER THEN ERROR(44) ELSE
    BEGIN
      ADDRESSVAR(DOMAIN,LATTR) ; TRANSFER(LATTR,RP) ;
      IT1 := DOMAIN^.VTYPE^.SIZE ; PT := ELTYPE ;
    END ;
    IF PT = NIL THEN
    BEGIN SKIP(49);  GOTO 10  END;
    IF NO = 15 THEN  (* SY = , *)
    BEGIN
      PT := PT^.RECVAR;
      REPEAT
        IF PT = NIL THEN
        BEGIN ERROR(66);  SKIP(49);  GOTO 10  END;
        INSYMBOL;
        IF NO = 1 THEN
        BEGIN SEARCH;
          IF CTPTR = NIL THEN
          BEGIN ERROR(31);  SKIP(49);  GOTO 10 END;
          IF CTPTR^.KLASS # KONST THEN
          BEGIN ERROR(63);  SKIP(49);  GOTO 10 END;
          WITH PT^ DO
          IF (CASETYPE^.FORM = SYMBOLIC) ^
              (CTPTR^.CONTYPE # CASETYPE)  v
            (CASETYPE^.FORM = NUMERIC) ^
              (CTPTR^.CONTYPE # INTPTR) THEN
          BEGIN ERROR(73);  SKIP(49);  GOTO 10 END;
          IVAL := CTPTR^.VALUES;
        END ELSE
        IF (NO # 2) v (CL IN [2,3]) THEN
        BEGIN ERROR(63);  SKIP(49);  GOTO 10 END ELSE
        WITH PT^ DO
        IF (CL = 1) ^ (CASETYPE^.FORM # NUMERIC)
          v (CL = 4) ^ (CASETYPE # CHARPTR) THEN
        BEGIN ERROR(73);  SKIP(49);  GOTO 10 END;
        CTPTR := PT^.VARIANTS;  BT1 := FALSE;
        WHILE (CTPTR # NIL) ^ ~BT1 DO
        WITH CTPTR^ DO
          IF CASEVAL = IVAL THEN BT1 := TRUE
                            ELSE CTPTR := NXTEL;
        IF CTPTR = NIL THEN
        BEGIN IT := PT^.CASESIZE;  PT := NIL END ELSE
        BEGIN IT := CTPTR^.CASESIZE;  PT := CTPTR^.VARIANTS END;
        INSYMBOL;
      UNTIL NO # 15;
    END ELSE IT := PT^.SIZE;
    GEN30(72B,6,RP,IT) ; GEN15(54B,6,RP,0) ;
    GEN30(70B,0,RP,IT1) ; GEN15(37B,0,0,6) ;
    GEN15(10B,6,RP,0) ; GEN30(03B,2,0,IC + (CP+1) DIV 2) ;
    GEN15(54B,6,RP,0) ; GEN30(71B,6,0,NILVAL) ;
    NOOP ; RP := RP - 1 ; TRANSFER(GATTR,6) ;
    IF NO # 10 THEN  (* SY # ) *)
    BEGIN ERROR(48) ; SKIP(49) END ELSE INSYMBOL ;
```

```
0:    END {ALLO} ;

      PROCEDURE PCK ;
         VAR LPT : CTP ; LLO, LHI : SHRTINT ;
      BEGIN
         IF NO ≠ 9 THEN {SY ≠ (}
         BEGIN ERROR(79) ; SKIP(49) ; GOTO 10 END ;
         INSYMBOL ; VARIAB ;
         WITH GATTR DO
         IF TYPTR ≠ NIL THEN
         BEGIN
            WITH TYPTR↑ DO
            IF (FORM ≠ ARRAYS)∨(AELTYPE ≠ CHARPTR) THEN ERROR(44) ELSE
            BEGIN LPT := INXTYPE ; DPLMT := DPLMT + (ALFALENG-1) - LO ;
               LLO := LO ; LHI := HI - (ALFALENG - 1) ;
            END ;
            LOADADR(GATTR,1) ; RP := 1 ;
         END ELSE LPT := INTPTR;
         IF NO ≠ 15 THEN
         BEGIN ERROR(80) ; SKIP(49) ; GOTO 10 END ;
         INSYMBOL ; EXPRESSION ;
         WITH GATTR DO
         IF TYPTR ≠ NIL THEN
         BEGIN
            IF (TYPTR↑.FORM ≠ NUMERIC)∨(LPT↑.FORM ≠ NUMERIC) THEN
            BEGIN IF TYPTR ≠ LPT THEN ERROR(44) END ELSE
            BEGIN
               TRANSFER(GATTR,2) ; RP := 2 ;
               IF INXCHECK THEN CHECKBNDS(2,LLO,LHI,INXERR) ;
               GEN15(36B,0,1,2) ;
               GEN30(61B,7,0,-ALFALENG) ; GEN15(13B,6,0,0) ;
               NOOP ; IT := IC ;
               GEN30(61B,7,7,1) ; GEN15(53B,1,0,7) ;
               GEN15(20B,6,0,6) ; GEN15(12B,6,1,6) ;
               GEN30(05B,7,0,IT) ; RP := 0
            END ;
         END ;
         IF NO ≠ 15 THEN
         BEGIN ERROR(80) ; SKIP(49) ; GOTO 10 END ;
         INSYMBOL ; VARIAB ;
         IF GATTR.TYPTR ≠ NIL THEN
            IF GATTR.TYPTR ≠ ALFAPTR THEN ERROR(44)
                                     ELSE TRANSFER(GATTR,6) ;
         IF NO ≠ 10 THEN {SY ≠ )}
         BEGIN ERROR(48) ;SKIP(49) ; END ELSE INSYMBOL ;
```

```
08    END  (* PCK *)  ;

      PROCEDURE UNPCK ;
         VAR LPT : CTP ; LLO, LHI : SHRTINT ;
      BEGIN
         IF NO ≠ 9 THEN  (* SY ≠ ( *)
         BEGIN ERROR(79) ; SKIP(49) ; GOTO 10 END ;
         INSYMBOL ; EXPRESSION ;
         IF GATTR.TYPTR ≠ NIL THEN
            IF GATTR.TYPTR ≠ ALFAPTR THEN ERROR(44)
                                     ELSE TRANSFER(GATTR,1) ;
         IF NO ≠ 15 THEN
         BEGIN ERROR(80) ; SKIP(49) ; GOTO 10 END ;
         INSYMBOL ; VARIAB ;
         WITH GATTR DO
         IF TYPTR ≠ NIL THEN
         BEGIN
            WITH TYPTR↑ DO
            IF (FORM ≠ ARRAYS)∨(AELTYPE ≠ CHARPTR) THEN ERROR(44) ELSE
            BEGIN LPT := INXTYPE ; DPLMT := DPLMT + (ALFALENG-1) - LO ;
            LLO := LO ; LHI := HI - (ALFALENG - 1) ;
         END ;
            LOADADR(GATTR,2) ; RP := 2 ;
         END ELSE LPT := INTPTR;
         IF NO ≠ 15 THEN
         BEGIN ERROR(80) ; SKIP(49) ; GOTO 10 END ;
         INSYMBOL ; EXPRESSION ;
         WITH GATTR DO
         IF TYPTR ≠ NIL THEN
         BEGIN
            IF (TYPTR↑.FORM ≠ NUMERIC)∨(LPT↑.FORM ≠ NUMERIC) THEN
            BEGIN IF TYPTR ≠ LPT THEN ERROR(44) END ELSE
            BEGIN
               TRANSFER(GATTR,3) ; RP := 3 ;
               IF INXCHECK THEN CHECKBNDS(3,LLO,LHI,INXERR) ;
               GEN15(36B,2,2,3) ;
               GEN30(61B,7,0,-ALFALENG) ; GEN15(43B,0,0,54) ;
               NOOP ; IT := IC ;
               GEN30(61B,7,7,1) ; GEN15(20B,1,0,6) ;
               GEN15(15B,6,1,0) ; GEN15(53B,6,2,7) ;
               GEN30(05B,7,0,IT) ;
            END
      END ;
      IF NO ≠ 10 THEN  (* SY ≠ ) *)
      BEGIN ERROR(48) ;SKIP(49) ; END ELSE INSYMBOL ;
```

```
0#    END  { UNPCK } ;

      PROCEDURE INSAPP(LPSW : SHRTINT) ;
      VAR LVALUE : SHRTINT ; LOF : BOOLEAN ;
      BEGIN
         IF NO ≠ 9 THEN  { SY ≠ ( }
         BEGIN ERROR(79) ; SKIP(49) ; GOTO 10 END ;
         INSYMBOL ;
         IF LPSW = 8 THEN EXPRESSION ELSE VARIAB ;
         IF GATTR.TYPTR ≠ NIL THEN
            IF GATTR.TYPTR↑.FORM ≠ NUMERIC THEN ERROR(44)
                                           ELSE TRANSFER(GATTR,1) ;
         IF NO ≠ 15 THEN
         BEGIN ERROR(80) ; SKIP(49) ; GOTO 10 END ;
         INSYMBOL ; EXPRESSION ;
         LOF := FALSE ;
         WITH GATTR DO
         IF TYPTR ≠ NIL THEN
            IF TYPTR↑.FORM ≠ NUMERIC THEN ERROR(44) ELSE
            IF KIND = SVAL THEN
            BEGIN LOF := TRUE ; LVALUE := VAL ; RP := 2 END
            ELSE TRANSFER(GATTR,2) ;
         IF NO ≠ 15 THEN
         BEGIN ERROR(80) ; SKIP(49) ; GOTO 10 END ;
         INSYMBOL ;
         IF LPSW = 8 THEN VARIAB ELSE EXPRESSION ;
         IF GATTR.TYPTR ≠ NIL THEN
            IF GATTR.TYPTR↑.FORM ≠ NUMERIC THEN ERROR(44)
                                       ELSE TRANSFER(GATTR,3) ;
         IF LOF THEN
         BEGIN IF LVALUE ≠ 0 THEN GEN15(20B,1,0,LVALUE) END ELSE
         BEGIN GEN15(63B,7,2,0) ; GEN15(22B,1,7,1) END ;
         GEN15(12B,6,1,3) ;
         IF LPSW = 8 THEN GEN15(54B,6,3,0) ELSE GEN15(54B,6,1,0) ;
         IF NO ≠ 10 THEN  { SY ≠ ) }
         BEGIN ERROR(48) ; SKIP(49) ; END ELSE INSYMBOL ;
```

```
08:   END {INSAPP} ;

      PROCEDURE READIR;
         CONST RD = 110B ;
      BEGIN IF NO ≠ 9 THEN {(} ;
         BEGIN ERROR(79);  SKIP(49);  GOTO 10  END;
         REPEAT
            INSYMBOL;  VARIAB;
            IF GATTR.TYPTR ≠ NIL THEN
               IF GATTR.TYPTR = CHARPTR THEN
               BEGIN {INLINE CODE}
                  GEN30(51B,2,0,INPT);  GEN15(53B,2,2,0);
                  GEN15(10B,6,2,0);  TRANSFER(GATTR,6);
                  GEN30(61B,7,0,INPT+1);  JUMPTO(GETC);
               END ELSE
               BEGIN LOADADR(GATTR,1) ;
                  IF GATTR.TYPTR↑.FORM = NUMERIC THEN
                     GEN15(66B,7,0,0)  ELSE
                  IF GATTR.TYPTR = REALPTR THEN
                     GEN30(61B,7,0,1)  ELSE ERROR(44) ;
                  JUMPTO(RD) ;
                  IF ASSCHECK THEN
                     IF (GATTR.TYPTR↑.FORM = NUMERIC)∧(GATTR.TYPTR ≠ INTPTR)
                     THEN CHECKBNDS(6,GATTR.TYPTR↑.MIN,GATTR.TYPTR↑.MAX,
                     ASSERR) ;
               END;  RP := 0;
         UNTIL NO ≠ 15 {,} ;
         IF NO ≠ 10 THEN {)}
         BEGIN ERROR(48);  SKIP(49)  END ELSE INSYMBOL ;
10:   END {READIR} ;

   PROCEDURE WRITEIR ;
      CONST WRTI=114B,WRTE=115B,WRTF=116B,WRTA=117B,WRTC=120B,
            WRTB=121B,WRTO=111B,TXT=112B;
      VAR LPTR : CTP ; LX2, FIN : BOOLEAN ;
          L,I, LCA, LCP : SHRTINT ; AWORD : ALFA ;
          A : ARRAY [1..ALFALENG] OF CHAR ;
          LIC : ADDRESS;

      PROCEDURE SETX2ANDJMP(FDEFVAL : INTEGER ; FADR : ADDRESS) ;
      BEGIN IF LX2 THEN GEN30(71B,2,0,FDEFVAL) ;
         JUMPTO(FADR)
      END ;

      PROCEDURE LOADEXP ;
      BEGIN INSYMBOL ; EXPRESSION ;
         IF GATTR.TYPTR ≠ NIL THEN
            IF GATTR.TYPTR↑.FORM ≠ NUMERIC THEN ERROR(62) ;
         TRANSFER(GATTR,RP)
      END ;

   BEGIN IF NO ≠ 9 THEN
      BEGIN ERROR(79) ; SKIP(49) ; GOTO 10 END ;
      REPEAT WHILE CH = ≡ ≡ DO NEXTCH ;
         IF CH = ≡≡≡≡ THEN
         BEGIN GEN30(04B,0,0,0) ; LCA := CA ; LCP := CP ;
            NOOP ; CODE[CA] := BUF ; I := 0 ; FIN := FALSE ;
            LIC := IC ; L := 0 ;
            REPEAT NEXTCH ;
               IF EOLFLAG THEN
               BEGIN ERROR(3) ; FIN := TRUE END ELSE
               BEGIN IF CH = ≡≡≡≡ THEN
                  BEGIN NEXTCH ; FIN := CH ≠ ≡≡≡≡ END ;
               IF ~FIN THEN
```

```
            BEGIN I := I + 1 ; A[I] := CH ; L := L + 1 ;
               IF I = ALFALENG THEN
               BEGIN PACK(A,1,AWORD) ; I := 0 ;
                 CA := CA + 1 ; IC := IC + 1 ;
                  IF CA > CODMAX THEN
                  BEGIN ERROR(90) ; CA := 1 END ;
                 CODE[CA] := ORD(AWORD)
               END ;
            END ;
         END ;
      UNTIL FIN ;
      FOR I := I + 1 TO ALFALENG DO A[I] := EOL ;
      PACK(A,1,AWORD) ; BUF := ORD(AWORD) ;
      CA := CA + 1 ; IC := IC + 1 ; INS(IC,LCP,LCA) ;
      IF CA > CODMAX THEN
      BEGIN ERROR(90) ; CA := 1 END ;
      INSYMBOL ;
      IF NO = 19 THEN ↑:↓
      BEGIN GEN30(71B,1,0,55B) ; RP := 1 ;
        LOADEXP ; GEN30(72B,2,2,-L) ;
        JUMPTO(WRTC)
      END ;
      GEN30(71B,1,0,LIC) ; JUMPTO(TXT)
   END ELSE
   BEGIN INSYMBOL ; EXPRESSION ; TRANSFER(GATTR,RP) ;
      LPTR := GATTR.TYPTR ;
      IF NO = 19 THEN ↑:↓
      BEGIN LOADEXP ; LX2 := FALSE ;
      END ELSE LX2 := TRUE ; ↑X2 NOT YET SET↓
      IF LPTR ≠ NIL THEN
      IF (NO=1)∧(AVAL=ΞOCTΞ)∧(LPTR↑.FORM≤POWER) THEN ↑O-FORMAT↓
      BEGIN SETX2ANDJMP(20,WRTO) ; INSYMBOL END ELSE
      IF LPTR↑.FORM = NUMERIC THEN SETX2ANDJMP(10,WRTI) ELSE
      IF LPTR = REALPTR THEN
      BEGIN IF NO = 19 THEN ↑:↓
         BEGIN LOADEXP ;
            SETX2ANDJMP(20,WRTF)
         END ELSE SETX2ANDJMP(20,WRTE) ;
      END ELSE
      IF LPTR = ALFAPTR THEN SETX2ANDJMP(10,WRTA) ELSE
      IF LPTR = BOOLPTR THEN SETX2ANDJMP(10,WRTB) ELSE
      IF LPTR = CHARPTR THEN
      BEGIN IF LX2 THEN ↑CALL PUTC ONLY↓
         BEGIN BXIXJ(6,1) ;
            GEN30(51B,1,0,OUTPT) ; GEN15(53B,6,1,0) ;
            GEN30(61B,7,0,OUTPT+1) ; JUMPTO(PUTC)
         END ELSE JUMPTO(WRTC)
      END ELSE ERROR(44) ;
   END ;
   RP := 0 ;
UNTIL NO ≠ 15 ; ↑,↓
IF NO ≠ 10 THEN ↑)↓
BEGIN ERROR(48) ; SKIP(49) END ELSE INSYMBOL ;
```

```
0:END ↑WRITEIR↓ ;

      PROCEDURE IFSTAT ;
      VAR LCA1, LCA2, LCP1, LCP2: SHRTINT ;
      BEGIN
         INSYMBOL ; EXPRESSION ;
         IF GATTR.TYPTR ≠ NIL THEN GENJP(0) ;
         LCA1 := CA ; LCP1 := CP ;
         IF NO ≠ 24 THEN  ↑ SY ≠ THEN ↓
         BEGIN
            IF GATTR.TYPTR ≠ NIL THEN ERROR(56) ; SKIP(24) ;
            IF NO ≠ 24 THEN
            BEGIN
               IF GATTR.TYPTR = NIL THEN ERROR(56) ;
               IF ERRCL[NO] = ENDSY THEN GOTO 30 ; GOTO 20 ;
            END
         END ;
        INSYMBOL ;
```

```
0:      STATEMENT ;
```

```
0:      IF NO ≠ 25 THEN ↱ SY ≠ ELSE ↳
        BEGIN NOOP;  INS(IC,LCP1,LCA1)  END ELSE
        BEGIN
           GEN30(048,0,0,0) ;
           LCA2 := CA ; LCP2 := CP ;
           NOOP ;
           INS(IC,LCP1,LCA1);
           INSYMBOL ; STATEMENT ;
           NOOP;  INS(IC,LCP2,LCA2);
        END
     END  ↱ IFSTAT ↳  ;


 PROCEDURE CASESTAT ;
 TYPE PTR = ↑LCSLABS ;
 VAR LCSLABS : CLASS   30   OF PACKED RECORD NEXT : PTR ;
                                              CSLAB : SHRTINT ;
                                              ADDR : ADDRESS ;
                                           END ;
     LCA, LCP, LCA1, LCP1, LCA2, LCP2, LMIN, LMAX : SHRTINT ;
       LPTR, PT1, PT2 : PTR ; LCTP : CTP ; LERR : BOOLEAN ;

   PROCEDURE ACASE ;
   VAR LCA,LCP : SHRTINT ; LERRFG : BOOLEAN ; PT1,PT2,PT3 : PTR ;
   BEGIN LERRFG := TRUE ; NOOP ;
     REPEAT IF NO IN [15,27] THEN ↱, OF↳ INSYMBOL ;
       IF NO = 1 THEN ↱ID↳
       BEGIN SEARCH ;
         IF CTPTR = NIL THEN
         BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
         WITH CTPTR↑ DO
         BEGIN IF KLASS ≠ KONST THEN
           BEGIN IF LERRFG THEN ERROR(61) ; GOTO 1 END ;
           IF CONTYPE ≠ NIL THEN
           BEGIN
             IF LCTP = NIL THEN LCTP := CONTYPE ELSE
             IF LCTP ≠ CONTYPE THEN ERROR(73) ;
             IT := VALUES ;
           END
         END
       END ↱IF NO = 1↳ ELSE
       IF NO = 2 THEN ↱CONST↳
       BEGIN CASE CL OF
         1: PT := INTPTR ;
         2: PT := REALPTR ;
         3: PT := ALFAPTR ;
         4: PT := CHARPTR ;
         END ;
         IF LCTP = NIL THEN LCTP := PT ELSE
         IF LCTP ≠ PT THEN ERROR(73) ;
         IT := IVAL ;
       END ↱IF NO = 2↳ ELSE
       BEGIN IF LERRFG THEN ERROR(61) ; GOTO 1 END ;
       IF IT ≥ TWOTO17 THEN ERROR(100) ;
       LERRFG := FALSE ; PT1 := LPTR ; PT2 := NIL ;
       IF ¬ERR THEN
       WHILE PT1 ≠ NIL DO
       BEGIN IF PT1↑.CSLAB ≥ IT THEN
         BEGIN IF PT1↑.CSLAB = IT THEN ERROR(77) ; GOTO 3 END ;
         PT2 := PT1 ; PT1 := PT1↑.NEXT ;
       END ;
       LMAX := IT ;
3:     NEW(PT3) ;
       IF PT3 ≠ NIL THEN
```

```
  BEGIN WITH PT3↑ DO
    BEGIN NEXT ↓= PT1 ; CSLAB ↓= IT ; ADDR ↓= IC END ;
    IF PT2 = NIL THEN
    BEGIN LPTR ↓= PT3 ; LMIN ↓= IT END ELSE PT2↑.NEXT ↓= PT3 ;
  END ELSE ERROR(71) ;
  INSYMBOL ;
UNTIL NO ≠ 15 ; ↑,↓
IF NO ≠ 19 THEN ↑↓ ERROR(64) ELSE INSYMBOL ;
```

```
;     STATEMENT ; GEN30(04B,0,0,0) ; LCA := CA ; LCP := CP ;
      IF NO = 16 THEN ↑;↓
      BEGIN INSYMBOL ; IF NO ≠ 22 THEN ACASE END ELSE
      IF ERRCL[NO] = BEGSY THEN
      BEGIN ERROR(58) ; GOTO 1 END ELSE
      IF NO ≠ 22 THEN ERROR(63) ;
      INS(IC+LMAX-LMIN+1,LCP,LCA) ;
    END ↑ACASE↓ ;

  BEGIN ↑CASESTAT↓
    LMIN := 0 ; LMAX := 0 ; LPTR := NIL ; LERR := ERR ; ERR := FALSE ;
    INSYMBOL ; EXPRESSION ;
    LCTP := GATTR.TYPTR ;
    IF LCTP ≠ NIL THEN
    BEGIN IF LCTP↑.FORM = NUMERIC THEN LCTP := INTPTR ELSE
      IF (LCTP↑.FORM > SYMBOLIC)∨(LCTP = REALPTR)∨(LCTP = ALFAPTR) THEN
      BEGIN ERROR(62) ; LCTP := NIL END ;
      TRANSFER(GATTR,RP) ;
    END ;
    IF INXCHECK THEN
    BEGIN GEN30(71B,7,0,IC) ; GEN30(71B,2,0,0) ;
      LCA1 := CA ; LCP1 := CP ; GEN15(37B,0,1,2) ;
      GEN30(71B,2,0,0) ; LCA2 := CA ; LCP2 := CP ;
      GEN15(37B,2,2,1) ; GEN15(12B,2,2,0) ;
      GEN30(03B,3,2,INXERR)
    END ;
    GEN15(63B,7,1,0) ; GEN30(02B,7,0,0) ; LCA := CA ; LCP := CP ;
    IF NO ≠ 27 THEN ↑OF↓
    BEGIN IF LCTP ≠ NIL THEN ERROR(65) ; SKIP(27) ;
      IF NO ≠ 27 THEN
      IF LCTP = NIL THEN ERROR(65) ;
    END ;
    ACASE ; INS(IC-LMIN,LCP,LCA) ; PT1 := LPTR ;
    IF INXCHECK THEN
    BEGIN INS(LMIN,LCP1,LCA1) ; INS(LMAX,LCP2,LCA2) ; END ;
    IT := LMIN ; IT1 := IC + LMAX - LMIN + 1 ; NOOP ;
    IF ERR THEN
    WHILE PT1 ≠ NIL DO
    BEGIN WHILE PT1↑.CSLAB ≠ IT DO
      BEGIN IF INXCHECK THEN
        BEGIN GEN30(71B,7,0,IC) ; GEN30(04B,0,0,INXERR) END ELSE
        BEGIN GEN30(04B,0,0,IT1) ; NOOP END ;
        IF ERR THEN GOTO 10 ;
        IT := IT + 1
      END ;
      GEN30(04B,0,0,PT1↑.ADDR) ; NOOP ;
      PT1 := PT1↑.NEXT ; IT := IT + 1 ;
    END ;
```

```
0: IF NO = 22 THEN ↑END↓ INSYMBOL ;
   ERR := ERR ∨ LERR ;
END ↑CASESTAT↓ ;

    PROCEDURE REPEATSTAT ;
    VAR LJPADDR: ADDRESS ;
    BEGIN
       NOOP ; LJPADDR := IC ;
       REPEAT
       BEGIN
          INSYMBOL ;
```

```
0:          STATEMENT ;
```

```
      IF ERRCL[NO] = BEGSY THEN BEGIN ERROR(58) ; GOTO 20 END ;
      IF NO = 25 THEN  ↱ELSE↓
      BEGIN ERROR(54) ; INSYMBOL ; GOTO 20 END
   END
   UNTIL NO ≠ 16 ;  ↱ SY ≠ ; ↓
   IF NO ≠ 29 THEN  ↱ SY ≠ UNTIL ↓  ERROR(67) ELSE
   BEGIN
      INSYMBOL ; EXPRESSION ;
      IF GATTR.TYPTR ≠ NIL THEN GENJP(LJPADDR) ELSE SKIP(49)
   END
END  ↱ REPEATSTAT ↓  ;

PROCEDURE WHILESTAT ;
VAR LJPADDR : ADDRESS;  LCA,LCP : SHRTINT;
BEGIN
   NOOP ; LJPADDR := IC ;
   INSYMBOL ; EXPRESSION ;
   IF GATTR.TYPTR ≠ NIL THEN GENJP(0) ;
   LCA := CA ; LCP := CP ;
   IF NO ≠ 31 THEN  ↱ SY ≠ DO ↓
   BEGIN
      IF GATTR.TYPTR ≠ NIL THEN ERROR(59) ; SKIP(31) ;
      IF NO ≠ 31 THEN
      BEGIN
         IF GATTR.TYPTR = NIL THEN ERROR(59) ;
         IF ERRCL[NO] = BEGSY THEN GOTO 20 ; GOTO 10
      END
   END ;
   INSYMBOL ;
```

```
      GEN30(048,0,0,LJPADDR) ;

0:    NOOP;  INS(IC,LCP,LCA);
    END  { WHILESTAT } ;

    PROCEDURE FORSTAT ;
    VAR LATTR : ATTR ; LCLASS, LCA, LCP : SHRTINT ;
      LJPADDR : ADDRESS ; LOF : BOOLEAN ; LUPBND : SHRTINT ;

    PROCEDURE CHTYPE ;
    BEGIN
      WITH GATTR DO
      IF TYPTR ≠ NIL THEN
      IF (TYPTR↑.FORM > SYMBOLIC)∨(TYPTR = REALPTR)∨
      (TYPTR = ALFAPTR) THEN
      BEGIN ERROR(62) ; TYPTR := NIL END
    END ;

    PROCEDURE CHTYPES ;
    BEGIN
      WITH GATTR DO
      IF (TYPTR ≠ NIL)∧(LATTR.TYPTR ≠ NIL) THEN
      IF((TYPTR↑.FORM = SYMBOLIC)∨(LATTR.TYPTR↑.FORM = SYMBOLIC))∧
      (TYPTR ≠ LATTR.TYPTR) THEN
      BEGIN ERROR(73) ; TYPTR := NIL END
    END ;

    BEGIN LCA := 0 ;
      INSYMBOL ;
      IF NO ≠ 1 THEN
      BEGIN ERROR(49) ; GATTR.TYPTR := NIL END ELSE
      BEGIN SEARCH ;
        IF CTPTR = NIL THEN
        BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
        IF CTPTR↑.KLASS ≤ PROC THEN
        BEGIN ERROR(32) ; INSYMBOL END ELSE VARIABLE
      END ;
      CHTYPE ;
      IF GATTR.TYPTR ≠ NIL THEN
      WITH GATTR DO
      IF ACCESS ≠ DRCT THEN
      BEGIN ERROR(69) ; TYPTR := NIL END ELSE
      IF ¬(BREG IN [0,LEVEL]) THEN ERROR(69) ;
      LATTR := GATTR ;
      IF NO ≠ 20 THEN  { SY ≠ := }
      BEGIN
        IF GATTR.TYPTR ≠ NIL THEN ERROR(52) ; SKIP(20);
        IF NO ≠ 20 THEN
        BEGIN
          IF GATTR.TYPTR = NIL THEN ERROR(52) ;
          IF ERRCL[NO] = BEGSY THEN GOTO 20 ; GOTO 10
        END
      END ;
      INSYMBOL ; EXPRESSION ;
      CHTYPE ; CHTYPES ;
      TRANSFER(GATTR,RP) ;
      IF NO ≠ 33 THEN  { SY ≠ TO/DOWNTO }
      BEGIN
        IF GATTR.TYPTR ≠ NIL THEN ERROR(70) ; SKIP(33) ;
        IF NO ≠ 33 THEN
        BEGIN
          IF GATTR.TYPTR = NIL THEN ERROR(70) ;
          IF ERRCL[NO] = BEGSY THEN GOTO 20 ; GOTO 10
        END
      END ;
      LCLASS := CL ;
```

```
INSYMBOL ; EXPRESSION ;
CHTYPE ; CHTYPES ;
IF (GATTR.TYPTR ≠ NIL)^(LATTR.TYPTR ≠ NIL) THEN
BEGIN
   TRANSFER(GATTR,RP) ;
   IF GATTR.KIND = SVAL THEN
   BEGIN LOF := TRUE ; LUPBND := GATTR.VAL END ELSE
   BEGIN GEN15(10B,7,RP,0) ;
      GEN30(51B,7,5,LC+TCT) ; TCT := TCT + 1 ;
      IF TCT > TMAX THEN TMAX := TCT ; LOF := FALSE ;
   END ;
   GEN15(10B,6,1,0) ; NOOP ; LJPADDR := IC ;
   IF LCLASS = 1 THEN ↑STEP +1↓ GEN15(37B,1,2,6)
                 ELSE ↑STEP -1↓ GEN15(37B,1,6,2) ;
   GEN30(03B,3,1,0) ; LCA := CA ; LCP := CP ; RP := 0 ;
   IF ASSCHECK THEN
   WITH LATTR.TYPTR↑ DO
   IF FORM = NUMERIC THEN
   BEGIN IF LATTR.TYPTR ≠ INTPTR THEN
      CHECKBNDS(6,MIN,MAX,ASSERR)
   END ELSE CHECKBNDS(6,0,FCONST↑.VALUES,ASSERR) ;
   TRANSFER(LATTR,6) ;
END ;
IF NO ≠ 31 THEN  ↑ SY ≠ DO ↓
BEGIN
   IF GATTR.TYPTR ≠ NIL THEN ERROR(59) ; SKIP(31) ;
   IF NO ≠ 31 THEN
   BEGIN
      IF GATTR.TYPTR = NIL THEN ERROR(59) ;
      IF ERRCL[NO] = BEGSY THEN GOTO 20 ; GOTO 10
   END
END ;
INSYMBOL ;
```

```
0:      STATEMENT ;
```

```
IF LATTR.TYPTR ≠ NIL THEN
BEGIN TRANSFER(LATTR,RP) ;
   WITH LATTR.TYPTR↑ DO
   IF (FORM = NUMERIC)∧((ABS(MIN) ≥ TWOTO17)∨(ABS(MAX) ≥
   TWOTO17)) THEN
   BEGIN GEN30(71B,0,0,1) ;
      GEN15(35B+LCLASS,6,1,0) ;  ↑ +1 OR -1↓
   END ELSE GEN30(72B,6,1,3-2*LCLASS) ;
   IF LOF THEN GEN30(71B,2,0,LUPBND) ELSE
   BEGIN TCT := TCT - 1 ; GEN30(51B,2,5,LC+TCT) END ;
   GEN30(04B,0,0,LJPADDR) ; NOOP ;
   INS(IC,LCP,LCA);
END ;
```

```
0:    END  * FORSTAT *  ;

      PROCEDURE GOTOSTAT ;
      BEGIN INSYMBOL ;
            IF (NO = 1)^(AVAL = ΞEXITΞ) THEN
            BEGIN INSYMBOL ;
               IF (NO ≠ 2)∨(CL ≠ 1) THEN
               BEGIN ERROR(61) ; SKIP(49) END ELSE
               BEGIN FOR IT := CEXTABIX DOWNTO 1 DO
                  IF EXTAB[IT].EXVAL = IVAL THEN
                  BEGIN IT1 := EXTAB[IT].JMPTABIX ;
                     IF JMPTAB[IT1] ≠ 0 THEN
                     BEGIN IF JMPTAB[IT1] ≠ LEVEL THEN
                        BEGIN LOADBASE(1,JMPTAB[IT1]) ; GEN15(63B,5,1,0)
                        END
                     END ELSE
                     BEGIN GEN30(51B,1,0,PARMLIST.IC0) ;
                        GEN15(63B,5,1,0)
                     END ;
                     GEN30(04B,0,0,PARMLIST.LPJMPTAB+IT1) ; GOTO 1
                  END ;
                  ERROR(43) ;
```

```
            INSYMBOL
         END
      END ELSE
      IF (NO ≠ 2)∨(CL ≠ 1) THEN
      BEGIN ERROR(61) ; SKIP(49) END ELSE
      BEGIN IF IVAL ≥ TWOTO17 THEN
         BEGIN ERROR(100) ; GOTO 20 END ;
          ↱ SEARCH THROUGH LABELTABLE OF CURRENT BLOCK ↳
         FOR IT := 1 TO CLABIX DO
         WITH LABTAB[IT] DO
         BEGIN
            IF LABVAL = IVAL THEN  ↱ LABEL ALREADY OCCURED ↳
            BEGIN
                ↱ IF DECL. OCC. GENERATE CODE ELSE CHAIN OCC. ↳
               IF FLD2 = 0 THEN GEN30(04B,0,0,FLD3) ELSE
               BEGIN
                  GEN30(04B,0,0,0) ;
                  IF CHNIX = 0 THEN
                  BEGIN ERROR(75) ; GOTO 20 END ;
                  WITH UNDLAB[CHNIX] DO
                  BEGIN
                     IT1 := SUCC ; SUCC := FLD3 ;
                     FLD3 := CHNIX ; PLACE := CA ;
                     LFTSH := CP ;
                  END ;
                  CHNIX := IT1
               END ;
               GOTO 20
            END
         END ;
          ↱ LABEL NOT YET MET, ENTER IT INTO LABELTABLE ↳
         GEN30(04B,0,0,0) ;
         IF CLABIX = MAXLABS THEN BEGIN ERROR(74) ; GOTO 20 END ;
         IF CHNIX = 0 THEN BEGIN ERROR(75) ; GOTO 20 END ;
         CLABIX := CLABIX + 1 ;
         WITH LABTAB[CLABIX], UNDLAB[CHNIX] DO
         BEGIN
            IT1 := SUCC ; LABVAL := IVAL ;
            SUCC := 0 ; FLD2 := CHNIX ;
            FLD3 := CHNIX ; PLACE := CA ;
            LFTSH := CP
         END ;
         CHNIX := IT1 ;
```

```
0:          INSYMBOL ;
```

```
      END ;
END  { GOTOSTAT }  ;

PROCEDURE WITHSTAT ;
VAR LTPCT, LB7CT: SHRTINT ;
BEGIN
   LTPCT := 0 ; LB7CT := 0 ;
   REPEAT
      INSYMBOL ;
      VARIAB ;
      WITH GATTR DO
      IF TYPTR ≠ NIL THEN
      IF TYPTR↑.FORM ≠ RECORDS THEN ERROR(38) ELSE
      BEGIN
         TOP := TOP + 1 ; LTPCT := LTPCT + 1 ;
         IF TOP > DISPLIMIT THEN ERROR(82) ELSE
         WITH DISPLAY[TOP] DO
         BEGIN
            FNAME := TYPTR↑.FSTFLD ;
            IF (ACCESS = DRCT)∧(BREG IN [0,LEVEL]) THEN
            BEGIN OCCUR := CWITH;
               CDSPL := DPLMT;  CLEV := BREG;
            END ELSE
            BEGIN
               LOADADR(GATTR,6) ; IT := LC + TCT ;
               GEN30(51B,6,5,IT) ;
               OCCUR := VWITH ; VDSPL := IT ;
               TCT := TCT + 1 ; LB7CT := LB7CT + 1 ;
               IF TCT > TMAX THEN TMAX := TCT ;
            END
         END
      END
   UNTIL NO ≠ 15 ;  { SY ≠ , }
   IF NO ≠ 31 THEN
   BEGIN
      IF GATTR.TYPTR ≠ NIL THEN ERROR(59) ; SKIP(31) ;
      IF NO ≠ 31 THEN
      BEGIN
         IF GATTR.TYPTR = NIL THEN ERROR(59) ;
      IF ERRCL[NO] = BEGSY THEN GOTO 20 ; GOTO 10
      END
   END ;
   INSYMBOL ;
```

```
0:      TOP := TOP - LTPCT ; TCT := TCT - LB7CT ;
     END  { WITHSTAT }  ;

{ ------------------------------------------------------------------ }

 PROCEDURE COMPSTAT ;
 BEGIN REPEAT BEGIN
       INSYMBOL ;
```

```
*       STATEMENT ;
        IF ERRCL[NO] = BEGSY THEN
      BEGIN ERROR(58) ; GOTO 1 END ;
      IF NO = 25 THEN  ↑ELSE↓
      BEGIN ERROR(54) ; INSYMBOL ; GOTO 1 END
      END
    UNTIL NO ≠ 16 ↑;↓ ;
    IF NO ≠ 22 THEN ↑END↓ ERROR(68) ELSE INSYMBOL ;
  END ↑COMPSTAT↓ ;

  PROCEDURE STATEMENT ;
     VAR LPSW : SHRTINT;
  BEGIN
    IF (NO = 2)^(CL = 1) THEN ↑LABEL↓
    BEGIN NOOP ;
      IF IVAL ≥ TWOTO17 THEN
      BEGIN ERROR(100) ; GOTO 1 END ;
      FOR IT := 1 TO CLABIX DO
      WITH LABTAB[IT] DO
      IF LABVAL = IVAL THEN ↑FOUND↓
      BEGIN IF FLD2 = 0 THEN ↑MULTIDEF.↓ ERROR(77) ELSE ↑FIXUP↓
        BEGIN IT1 := FLD3 ;
          REPEAT WITH UNDLAB[IT1] DO
            BEGIN INS(IC,LFTSH,PLACE) ; IT1 := SUCC END
          UNTIL IT1 = 0 ;
          IT1 := FLD2 ; UNDLAB[IT1].SUCC := CHNIX ;
          CHNIX := FLD3 ; FLD2 := 0 ; FLD3 := IC ;
        END ;
        GOTO 1
      END ↑IF, WITH, FOR↓ ;
      ↑NEW LABEL↓
      IF CLABIX = MAXLABS THEN ERROR(74) ELSE
      BEGIN CLABIX := CLABIX + 1 ;
        WITH LABTAB[CLABIX] DO
        BEGIN LABVAL := IVAL ; FLD2 := 0 ; FLD3 := IC END ;
      END ;
```

```
*     INSYMBOL ;
      IF NO ≠ 19 THEN ↱!↴
      BEGIN ERROR(64) ; SKIP(49) END ELSE INSYMBOL ;
    END ↱IF (NO=2)^(CL=1)↴ ;
    RP := 0 ;
    CASE SPLITSTAT[NO] OF
    ↱PASS  ↴  1:  ↱ENDSY OR IRRELSY↴ ;
    ↱IDENT ↴  2:  BEGIN SEARCH ;
                    IF CTPTR = NIL THEN
                    BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
                    WITH CTPTR↑ DO
                    IF KLASS ≤ KONST THEN ERROR(55) ELSE
                    IF (KLASS = PROC)^
                    ((PROCTYPE = CTPTR)v(PROCTYPE = NIL)) THEN
                    BEGIN ↱ PROCCALL ↴
                      IF PROCTYPE = CTPTR THEN
                      BEGIN INSYMBOL ; IF CTPTR ≤ PREDEFP THEN
                        BEGIN LPSW := CTPTR↑.SEGSIZE ;
                         CASE LPSW OF
           1,2,3,4,5:      GETPUT(LPSW) ;
                     6:      PCK ;
                     7:      UNPCK ;
                   8,9:      INSAPP(LPSW) ;
                    10:      READIR ;
                    11:      WRITEIR ;
                    12:      ALLC ;
                        END ;
                        END ELSE  PASSPARAMS ;
                      END ELSE SKIP(49)
                    END ↱PROCCALL↴ ELSE  ASSIGN ;
                  END ;
    ↱BEGIN ↴  3:  COMPSTAT ;
    ↱IF    ↴  4:  IFSTAT ;
    ↱CASE  ↴  5:  CASESTAT ;
    ↱REPEAT↴  6:  REPEATSTAT ;
    ↱WHILE ↴  7:  WHILESTAT ;
    ↱FOR   ↴  8:  FORSTAT ;
    ↱GOTO  ↴  9:  GOTOSTAT ;
    ↱WITH  ↴ 10:  WITHSTAT ;
    END ;
    IF ERRCL[NO] = IRRELSY THEN
    BEGIN ERROR(54) ; SKIP(49) END ;
    RP := 0
  END ↱STATEMENT↴ ;

↱ ------------------------------------------------------------------ ↴

  PROCEDURE TYPEDECL(VAR TL:SHRTINT; VAR P1:CTP);
  VAR I,L,LL,J,CV,E1,E2,BDISPL : SHRTINT;
     OPT : OPTPWR;  DISPL : ADDRESS;  PACKFLAG,LERR : BOOLEAN;
     LASTFLD,PP,P,NXTF,NXTC,NXTA,RTYP : CTP;

     PROCEDURE SKIPT(FNO : SHRTINT) ;
     BEGIN
        WHILE (TERRCL[NO] = IRRELSY) ^ (FNO ≠ NO) DO INSYMBOL;
     END ↱SKIPT↴;

     PROCEDURE TYPERR(I : SHRTINT);
     BEGIN TL := 0;  P1 := NIL;
        ERROR(I);  SKIPT(49);
     END ↱TYPERR↴;

     PROCEDURE SUBRANGE(VAR VAL1,VAL2 : INTEGER; VAR N1 : CTP;
        P : CTP) ;
```

```
      (*THE FIRST SYMBOL OF SUBRANGE HAS BEEN READ.
         THE PROCEDURE RETURNS THE TWO BOUND-VALUES IN VAL1,VAL2,
         AND RETURNS  N1 = POINTER TO TYPE OF CONSTANTS.
         ERRORS : TYPES DO NOT AGREE, TYPE IS NOT INTEGER,CHAR,
            OR SYMBOLIC,  VAL1 > VAL2.
         P INDICATES BEGINNING OF SEARCH FOR FIRST SYMBOL IF IT IS
         AN ID.   *)
      VAR N2 : CTP;
   BEGIN INCONST(VAL1,N1,P);
      IF (N1=NIL) v (N1=REALPTR) v (N1=ALFAPTR) THEN
         ERR := TRUE ELSE
      BEGIN IF NO ≠ 19 THEN ERROR(10) ELSE INSYMBOL ;
         INCONST(VAL2,N2,NEXT);
         IF N1 ≠ N2 THEN ERR := TRUE ELSE
         IF VAL1 > VAL2 THEN ERROR(25);
      END;
   END (*SUBRANGE*);

   PROCEDURE SUBTYPE(VAR I,J : SHRTINT; VAR P : CTP);
      (*EITHER A TYPE-ID FOR A SUBRANGE OR AN EXPLICIT SUBRANGE
         ARE PROCESSED.
         RETURNS I = LOWBOUND, J = HIGHBOUND,
            P = POINTER TO TYPE OF CONSTANTS  *)
   BEGIN
      IF NO = 1 THEN
      BEGIN SRCHREC(NEXT);
         IF CTPTR = NIL THEN SEARCH;
         IF CTPTR = NIL THEN ERROR(12) ELSE
         BEGIN IF CTPTR↑.KLASS = TYPES THEN
            BEGIN IF CTPTR↑.FORM > SYMBOLIC THEN ERROR(13) ELSE
               BEGIN
                 CASE CTPTR↑.FORM OF
NUMERIC:           BEGIN I := CTPTR↑.MIN;  J := CTPTR↑.MAX;
                     P := INTPTR;
                   END;
SYMBOLIC:          BEGIN IF (CTPTR = REALPTR)v(CTPTR = ALFAPTR) THEN
                     ERROR(6) ELSE
                     BEGIN I := 0 ; J := CTPTR↑.FCONST↑.VALUES ;
                       P := CTPTR ;
                     END
                   END;
                 END (*CASE*);
                 INSYMBOL;
               END;
            END (*TYPES* ELSE
            IF CTPTR↑.KLASS = KONST THEN
               SUBRANGE(I,J,P,CTPTR) ELSE ERROR(63) ;
         END;
      END (*ID* ELSE
      IF NO IN [2,7] THEN SUBRANGE(I,J,P,NIL) ELSE
        ERROR(1) ;
   END (*SUBTYPE*);

   PROCEDURE SCALDECL(N : CTP);
      VAR JJ : INTEGER ;
   BEGIN SUBRANGE(I,J,PP,N);
      IF ~ERR THEN
      IF PP↑.FORM = SYMBOLIC THEN
      BEGIN ERROR(28);  P1 := NIL  END ELSE
      BEGIN NEW(P,TYPES,NUMERIC);
         WITH P↑ DO
         BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
            SIZE := 1;  FORM := NUMERIC;
            MIN := I;  MAX := J;
            IF I ≥ 0 THEN BITS := LOG2PLUS1(J) ELSE
            BEGIN JJ := ABS(J) ;
```

```
                IF JJ < -I THEN JJ := -I ;
                BITS := LOG2PLUS1(JJ) + 1 {SIGN BIT}
             END
          END ;
          TL := 1;  P1 := P;
       END ELSE P1 := NIL
    END {SCALDECL};

PROCEDURE FIELDLIST(VAR MAXSIZE:SHRTINT; VAR VARPTR,NXTF:CTP);
    VAR MXL,L,LL,MINSIZE,CASEBITS,I : SHRTINT;
       P,PP,PP1,PP2,NXTC,NXT,CPTR : CTP;
       TAGFLAG,REP : BOOLEAN;

    PROCEDURE RECERR(I : SHRTINT);
    BEGIN ERROR(I);  SKIPT(49); END;

    PROCEDURE ADJUST;
       { MOVE LAST FIELD TO RIGHT. IF IT IS THE ONLY FIELD,
          CHANGE TO NONPACKED.
          IF LAST FIELD IS TAGFIELD THEN DO NOT MOVE.
          INCREASE DISPL, RESET BDISPL  }
    BEGIN IF ~TAGFLAG THEN
       WITH LASTFLD↑ DO
       BEGIN IF BITDISPL = 0 THEN {ONLY ONE FIELD IN WORD}
             BITWIDTH := 0 ELSE
             BITWIDTH := WORDLENGTH - BITDISPL;
       END;
       DISPL := DISPL + 1;  BDISPL := 0;
    END {ADJUST};

BEGIN
    TAGFLAG := TRUE;     NXT := NXTF;
    REPEAT
       IF NO = 26 THEN {CASE} GOTO 2 ;
       I := 0;
1:        IF NO ≠ 1 THEN
          BEGIN RECERR(11);
             IF TERRCL[NO] = BEGSY THEN GOTO 11;  GOTO 12;
          END;
          SRCHREC(NXT);
          IF CTPTR ≠ NIL THEN ERROR(5) ELSE
          BEGIN NEW(P,FIELD); I := I + 1;
             WITH P↑ DO
             BEGIN NAME := AVAL;  NXTEL := NXT;  KLASS := FIELD;
                FLDTYPE := NIL;
             END;  NXT := P;
          END;
          INSYMBOL;
          IF NO = 15 THEN {,}
          BEGIN INSYMBOL;  GOTO 1 END;
          IF NO ≠ 19 THEN {NOT :} ERROR(10) ELSE INSYMBOL;
11:       TYPEDECL(L,P);
          IF P ≠ NIL THEN
          IF P↑.FORM > RECORDS THEN ERROR(30) ELSE
          BEGIN IF PACKFLAG THEN
             BEGIN IF I > 1 THEN {REVERSE POINTERS}
                BEGIN PP := NXT;
                  FOR I := I DOWNTO 1 DO
                  BEGIN PP1 := PP↑.NXTEL;
                    PP↑.NXTEL := PP2;  PP2 := PP;
                    PP := PP1;
                  END;
                  NXT↑.NXTEL := PP;  NXT := PP2;
                END {REVERSE} ELSE PP := NXT↑.NXTEL;
                PP1 := NXT;
                IF P↑.FORM ≤ POWER THEN
```

```
                BEGIN
                  CASE P↑.FORM OF
NUMERIC:            LL := P↑.BITS;
SYMBOLIC:           LL := P↑.BITSIZE;
POINTER:            LL := 18 ;
POWER:              LL := P↑.PWBITS;
                  END;
                  REPEAT
                    IF BDISPL + LL > WORDLENGTH THEN ADJUST;
                    IF LL = WORDLENGTH THEN
                    BEGIN PP1↑.BITWIDTH := 0;
                      PP1↑.FLDADDR := DISPL;  DISPL := DISPL + 1;
                    END ELSE
                    BEGIN PP1↑.BITWIDTH := LL;
                      PP1↑.BITDISPL := BDISPL;
                      PP1↑.FLDADDR := DISPL;  BDISPL := BDISPL + LL;
                    END;
                    PP1↑.FLDTYPE := P;
                    LASTFLD := PP1;  TAGFLAG := FALSE;
                    PP1 := PP1↑.NXTEL;
                  UNTIL PP1 = PP;
                END ↑FORM ≤ POINTER↓ ELSE
                BEGIN IF BDISPL ≠ 0 THEN ADJUST;
                  TAGFLAG := FALSE;
                  REPEAT
                  PP1↑.BITWIDTH := 0;  PP1↑.FLDTYPE := P;
                    PP1↑.FLDADDR := DISPL;
                    DISPL := DISPL + L;  LASTFLD := PP1;
                    PP1 := PP1↑.NXTEL;
                  UNTIL PP1 = PP;
                END;
              END ↑PACKFLAG↓ ELSE
              BEGIN LL := DISPL + I*L;  DISPL := LL;
                PP := NXT;
                FOR I := I DOWNTO 1 DO
                BEGIN PP↑.FLDTYPE := P;  LL := LL - L;
                  PP↑.FLDADDR := LL;  PP↑.BITWIDTH := 0;
                  PP := PP↑.NXTEL;
                END;
              END;
            END ↑FORM ≤ RECORDS↓;
12:       IF NO = 16 THEN INSYMBOL;
        UNTIL (TERRCL[NO] = ENDSY) ^ (NO ≠ 26);
        IF BDISPL ≠ 0 THEN ADJUST;
        MAXSIZE := DISPL;  VARPTR := NIL;
        GOTO 9;
2:      ↑CASE↓
        INSYMBOL;
        IF NO ≠ 1 THEN ERROR(11) ELSE
        BEGIN SRCHREC(NXT);
          IF CTPTR ≠ NIL THEN ERROR(5) ELSE
          BEGIN NEW(P,FIELD);
            WITH P↑ DO
            BEGIN NAME := AVAL;  NXTEL := NXT;  KLASS := FIELD;
              FLDTYPE := NIL;
            END;  NXT := P;
          END;
          INSYMBOL;
        END ↑NO = 1↓;
          IF NO ≠ 19 THEN ERROR(10) ELSE INSYMBOL;
          IF NO ≠ 1 THEN ERROR(11) ELSE
          BEGIN SRCHREC(NEXT);
            IF CTPTR = NIL THEN SEARCH;
            IF CTPTR = NIL THEN ERROR(12) ELSE
            IF (CTPTR↑.KLASS≠TYPES) v (CTPTR↑.FORM>SYMBOLIC) THEN
              ERROR(7);
```

```
          INSYMBOL;
             IF NO ≠ 27 THEN ERROR(14);
             CPTR := CTPTR;
             IF CPTR ≠ NIL THEN
             IF PACKFLAG THEN
             BEGIN
                  CASE CPTR↑.FORM OF
NUMERIC:            LL := CPTR↑.BITS;
SYMBOLIC:           LL := CPTR↑.BITSIZE;
                  END;
                  IF BDISPL + LL > WORDLENGTH THEN ADJUST;
                  P↑.BITDISPL := BDISPL;  P↑.BITWIDTH := LL;
                  P↑.FLDADDR := DISPL;  BDISPL := BDISPL + LL;
               LASTFLD := P;  CASEBITS := BDISPL;
             END {PACKFLAG} ELSE
             BEGIN P↑.FLDADDR := DISPL;
                P↑.BITWIDTH := 0;
             END;
             P↑.FLDTYPE := CPTR;  DISPL := DISPL + 1;
             MINSIZE := DISPL;  MAXSIZE := DISPL;
             NXTC := NIL;  INSYMBOL;
             REPEAT  I := 0;
                REPEAT
                  IF (NO>2) ∨ (NO=2) ∧ ¬(CL IN [1,4]) THEN
                  BEGIN RECERR(63);  GOTO 3 END ;
                  IF CPTR ≠ NIL THEN
                    IF NO = 1 THEN
                    BEGIN SRCHREC(NEXT);
                      IF CTPTR = NIL THEN SEARCH;
                      IF CTPTR = NIL THEN ERROR(12) ELSE
                      WITH CTPTR↑ DO
                        IF KLASS ≠ KONST THEN ERROR(63) ELSE
                        IF (CPTR↑.FORM = SYMBOLIC) ∧
                            (CONTYPE ≠ CPTR)  ∨
                          (CPTR↑.FORM = NUMERIC) ∧
                            (CONTYPE ≠ INTPTR) THEN ERROR(73) ELSE
                        IT := VALUES;
                  END {NO = 1} ELSE
                  IF (CL = 1) ∧ (CPTR↑.FORM ≠ NUMERIC)
                    ∨ (CL = 4) ∧ (CPTR ≠ CHARPTR) THEN
                      ERROR(73) ELSE
                  IT := IVAL;  {END CPTR ≠ NIL}
                  NEW(P,TAGFIELD);
                  WITH P↑ DO
                  BEGIN NAME := BLANK;  NXTEL := NXTC;
                    KLASS := TAGFIELD;
                    TAGVAL := TRUE;  CASEVAL := IT;
                  END;  NXTC := P;  I := I + 1;
                  INSYMBOL;

                  IF NO = 15 THEN
                    BEGIN INSYMBOL ; REP := TRUE END
                 ELSE REP := FALSE;
          UNTIL ¬REP;
          IF NO ≠ 19 THEN ERROR(10) ELSE INSYMBOL;
          IF NO = 9 THEN  { ( FIELDLIST ) }
          BEGIN IF PACKFLAG THEN
            BEGIN DISPL := MINSIZE - 1;
              BDISPL := CASEBITS;
            END ELSE DISPL := MINSIZE;
            INSYMBOL;  FIELDLIST(MXL,PP,NXT);
            IF NO ≠ 10 THEN ERROR(9) ELSE INSYMBOL;
          END ELSE
          BEGIN PP := NIL;  MXL := MINSIZE END;
          P := NXTC;
          FOR I := I DOWNTO 1 DO
```

```
                WITH P↑ DO
                BEGIN CASESIZE := MXL;  VARIANTS := PP;
                  P := NXTEL;
                END;  IF MAXSIZE < MXL THEN MAXSIZE := MXL;
                IF NO = 16 THEN INSYMBOL;
              UNTIL NO > 2;
              NEW(P,TAGFIELD);
              WITH P↑ DO
              BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TAGFIELD;
                CASESIZE := MINSIZE;  VARIANTS := NXTC;
                TAGVAL := FALSE;  CASETYPE := CPTR;
              END;
              VARPTR := P;
        END ↑NO = 1↓;
9:    NXTF := NXT;
  END ↑ FIELDLIST ↓;

  ↑TYPEDECL↓
  BEGIN PACKFLAG := FALSE;
     IF NO = 42 THEN  ↑PACKED↓
     BEGIN PACKFLAG := TRUE;  INSYMBOL END;
     IF NO = 1 THEN
     BEGIN SRCHREC(NEXT);
        IF CTPTR = NIL THEN SEARCH;
        IF CTPTR = NIL THEN
        BEGIN ERROR(12);  P1 := NIL;  SKIPT(16) END ELSE
        BEGIN IF CTPTR↑.KLASS = TYPES THEN ↑TYPE-ID↓
           BEGIN TL := CTPTR↑.SIZE;  P1 := CTPTR;  INSYMBOL END ELSE
           IF CTPTR↑.KLASS = KONST THEN SCALDECL(CTPTR) ELSE
           TYPERR(11);
        END;
     END ↑ID↓ ELSE
     IF NO = 9 THEN  ↑ SYMBOLIC ↓
     BEGIN CV := 0; LERR := ERR ; ERR := FALSE;NEW(P,TYPES,SYMBOLIC);
        WITH P↑ DO
        BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
           FORM := SYMBOLIC;
        END;  RTYP := P;  NXTC := NIL;
        REPEAT INSYMBOL;
           IF NO ≠ 1 THEN
           BEGIN ERROR(11);  SKIPT(15);
              GOTO 2;
           END;
           SRCHREC(NEXT);
              IF CTPTR ≠ NIL THEN ERROR(3) ELSE
              BEGIN NEW(P,KONST);
                 WITH P↑ DO
                 BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := KONST;
                    CONTYPE := RTYP;
                    VALUES := CV;  SUCC := NXTC;
                 END;  CV := CV + 1;  NEXT := P;  NXTC := P;
              END;
              INSYMBOL;
2:      UNTIL NO ≠ 15;
           NEW(P,TYPES,POWER) ;
            WITH P↑ DO
            BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
               SIZE := 1;  FORM := POWER;
               ELSET := RTYP;  PWBITS := CV + 1;
            END;
            WITH RTYP↑ DO
            BEGIN FCONST := NEXT;  SIZE := 1;
               BITSIZE := LOG2PLUS1(CV - 1);  PWSET := P;
            END;
         IF ERR THEN P1 := NIL ELSE
         BEGIN ERR := LERR ; P1 := RTYP END ;
```

```
            TL := 1 ;
            IF NO ≠ 10 THEN TYPERR(9) ELSE INSYMBOL ;
         END  ↑ SYMBOLIC ↓  ELSE
         IF NO IN [2,7] THEN ↑ SUBRANGE ↓
         BEGIN LERR := ERR; ERR := FALSE;
            SCALDECL(NEXT);
            IF ERR THEN ERROR(6) ELSE ERR := LERR
         END ↑SUBRANGE↓ ELSE
         IF NO = 38 THEN  ↑ STRUCTURED TYPES ↓
         CASE CL OF
   ↑ARRAY↓ 1:
         BEGIN INSYMBOL;
            IF NO ≠ 11 THEN
            BEGIN TYPERR(98);
               IF TERRCL[NO] = BEGSY THEN TYPEDECL(I,CTPTR); GOTO 19;
            END;
               NXTA := NIL;
               REPEAT NEW(P,TYPES,ARRAYS);
                  WITH P↑ DO
                  BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
                     FORM := ARRAYS;  AELTYPE := NXTA;
                     ↑ AELTYPE TEMPORARILY LINKS SUBARRAYS ↓
                  END;  NXTA := P;
                  INSYMBOL;
                  LERR := ERR;  ERR := FALSE;
                  SUBTYPE(I,J,P);
                  IF ERR THEN
                  BEGIN ERROR(6);  SKIPT(15); I := 0; J := 0; P := NIL
                  END ELSE
                  ERR := LERR;
                  WITH NXTA↑ DO
                  BEGIN LO := I;  HI := J;  INXTYPE := P END;
               UNTIL NO ≠ 15;
               IF NO ≠ 12 THEN
               BEGIN ERROR(37); SKIPT(27);
                  IF TERRCL[NO] = BEGSY THEN GOTO 11;
                  IF NO = 27 THEN
                  BEGIN INSYMBOL; GOTO 11 END ;
                  IF NO ≠ 12 THEN
                  BEGIN P1 := NIL ; TL := 0; GOTO 19 END;
               END;
               INSYMBOL;
                  IF NO ≠ 27 THEN ERROR(14) ELSE INSYMBOL;
   11:            TYPEDECL(TL,CTPTR);
                  IF CTPTR ≠ NIL THEN
                  IF CTPTR↑.FORM > RECORDS THEN
                  BEGIN ERROR(30); CTPTR := NIL END ELSE
                  BEGIN
                     REPEAT
                       WITH NXTA↑ DO
                       BEGIN MULOPT(TL,E1,E2,OPT);
                         OPTTYP := OPT;  EXP1 := E1;  EXP2 := E2;
                         TL := TL*(HI - LO + 1);
                         SIZE := TL;
                         P := AELTYPE;  AELTYPE := CTPTR;
                       END;
                       CTPTR := NXTA;  NXTA := P;
                     UNTIL NXTA = NIL;
                     ↑ NOW TL = SIZE OF ARRAY, CTPTR POINTS TO IT ↓
                  END;  P1 := CTPTR;
   19:  END ↑ARRAY↓;
   ↑RECORD↓ 2:
         BEGIN NEW(P,TYPES,RECORDS);
            WITH P↑ DO
            BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
               FORM := RECORDS;
```

```
      END;  RTYP := P;
      INSYMBOL;  NXTF := NIL;
      DISPL := 0;  BDISPL := 0;  LERR := ERR;  ERR := FALSE;
      FIELDLIST(TL,P,NXTF);
      IF NO ≠ 22 THEN ERROR(17);
      IF ERR THEN TYPERR(18) ELSE
      WITH RTYP↑ DO
      BEGIN SIZE := TL;  FSTFLD := NXTF;  RECVAR := P;
         P1 := RTYP;  ERR := LERR;
      END;
      IF NO = 22 THEN INSYMBOL;
   END ↱RECORD↳;
↱FILE↳ 3:
   BEGIN INSYMBOL;
      IF NO = 27 THEN ↱OF↳
      BEGIN I := 8 ; INSYMBOL END ELSE
      BEGIN INCONST(I,PT,NEXT) ;
         IF (I≤0)∨(PT≠INTPTR) THEN
         BEGIN ERROR(95) ; I := 0 END ;
         IF NO ≠ 27 THEN ERROR(14) ELSE INSYMBOL ;
      END ;
      TYPEDECL(TL,CTPTR);
      IF CTPTR ≠ NIL THEN
      IF CTPTR↑.FORM > RECORDS THEN
      BEGIN ERROR(30); P1 := NIL END ELSE
      BEGIN L := CTPTR↑.SIZE;
         LL := L DIV 64 + 1 ;
         IF LL < I THEN LL := I ;
         LL := ((LL*64) DIV L + 1) * L ;
         NEW(P,TYPES,FILES);
         WITH P↑ DO
         BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
            FORM := FILES;  SIZE := LL;  FELTYPE := CTPTR;
         END;
         TL := LL;  P1 := P;
      END ELSE P1 := NIL;
   END ↱FILE↳;
↱CLASS↳ 4:
   BEGIN NEW(P,TYPES,CLASSS);
      WITH P↑ DO
      BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
         FORM := CLASSS;
      END;
      INSYMBOL ;
      IF NO = 27 THEN
      BEGIN I := 100 ; INSYMBOL END ELSE
      BEGIN INCONST(I,PT,NEXT) ;
         IF (I≤0)∨(PT≠INTPTR) THEN
         BEGIN ERROR(6) ; I := 0 END ;
         IF NO ≠ 27 THEN ERROR(14) ELSE INSYMBOL ;
      END ;
      TYPEDECL(TL,CTPTR);
      IF CTPTR ≠ NIL THEN
      IF CTPTR↑.FORM > RECORDS THEN
      BEGIN ERROR(30); P1 := NIL END ELSE
      BEGIN TL := TL*I + 1;
         P↑.SIZE := TL;  P↑.PELTYPE := CTPTR;
         P1 := P;
      END ELSE P1 := NIL ;
   END ↱CLASS↳;
↱SET↳ 5:
   BEGIN INSYMBOL;
      IF NO = 27 THEN INSYMBOL ELSE ERROR(14);
      LERR := ERR;  ERR := FALSE;
      SUBTYPE(I,J,P);
      IF ERR THEN TYPERR(6) ELSE
```

```
      BEGIN ERR := LERR;
         CASE P↑.FORM OF
NUMERIC:    IF (I < 0) ∨ (J > WORDLENGTH - 2) THEN
            BEGIN ERROR(6);  P1 := NIL END ELSE
               P1 := PNUMPTR;
SYMBOLIC:   BEGIN IF P = CHARPTR THEN J := WORDLENGTH - 2  ;
               IF J > WORDLENGTH - 2 THEN
               BEGIN ERROR(6); P1 := NIL END ELSE P1 := P↑.PWSET;
            END;
         END ↑CASE↓;
         TL := 1;
      END ↑ERR↓;
   END ↑SET↓;

   END ↑STRUCTURED TYPES↓ ELSE
   IF NO = 18 THEN ↑POINTER↓
   BEGIN INSYMBOL;
      IF NO ≠ 1 THEN TYPERR(11) ELSE
      BEGIN TL := 1;  NEW(P,TYPES,POINTER);
         WITH P↑ DO
         BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
            FORM := POINTER;  SIZE := 1;
         END;
         SRCHREC(NEXT);
         IF CTPTR = NIL THEN SEARCH;
         IF CTPTR ≠ NIL THEN
         IF CTPTR↑.VTYPE = NIL THEN CTPTR := NIL;
         IF CTPTR = NIL THEN ↑UNDECLARED CLASS↓
         IF PTX > PTLIMIT THEN
         BEGIN ERROR(92); P1 := NIL; INSYMBOL END ELSE
         WITH PTLIST[PTX] , P↑ DO
         BEGIN HNAME := AVAL;  PPTR := P;
            DOMAIN := P;  ELTYPE := P;
            PTX := PTX + 1;  P1 := P;  INSYMBOL;
         END ELSE
         WITH CTPTR↑ DO
         IF (KLASS = VARS) ^ (VTYPE↑.FORM = CLASSS) THEN
         BEGIN P↑.DOMAIN := CTPTR;
            P↑.ELTYPE := VTYPE↑.PELTYPE;
            P1 := P;  INSYMBOL;
         END ELSE TYPERR(15);
      END;
   END ↑POINTER↓ ELSE TYPERR(18);
END ↑TYPEDECL↓;

↑ ---------------------------------------------------------------- ↓

PROCEDURE BODY(SURRPTR,FIRSTENTRY:CTP);

LABEL 1;
CONST INP=0,PRINT=1,OUT=2,PUNCH=3,EXT=4,SCRATCH=5;

VAR I,J : INTEGER;
   OLDLEV : RG3;
   LC1,TL,LL,VLC,FSTIX,LCA,LCP : SHRTINT ;
   N,P,PROCPTR,LFIRSTENTRY : CTP;
   TYPID : ALFA ; LCH : CWORD ;

   PROCEDURE FINDSEMICOLON;
   BEGIN IF NO ≠ 16 THEN
      BEGIN ERROR(58);  SKIP(16);
         IF NO ≠ 16 THEN GOTO EXIT 1;
      END;  INSYMBOL;
   END ↑FINDSEMICOLON↓;

PROCEDURE VARDECL;
```

```
    LABEL 10;
    VAR AT : ADDRESS; I,J,FPX : SHRTINT;

    PROCEDURE FILERR;
    BEGIN ERROR(95);  SKIP(12);  GOTO EXIT 10 END;

BEGIN INSYMBOL;
   WHILE NO = 1 DO
   BEGIN I := 0;
      REPEAT
         SRCHREC(NEXT);
         IF CTPTR ≠ NIL THEN ERROR(8) ELSE
         BEGIN NEW(P,VARS);
            WITH P↑ DO
            BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := VARS;
               VKIND := ACTUAL;  VTYPE := NIL;
               VLEVEL := LEVEL ;
            END;
            NEXT := P;  I := I + 1;
         END;
         INSYMBOL;
         IF NO = 11 THEN ↱ [ IN,OUT,EXT,PRINT,PUNCH ] ↳
         IF LEVEL > 0 THEN FILERR ELSE
         BEGIN INSYMBOL;
           IF FILTOP = FILLIMIT THEN ERROR(92) ELSE
           BEGIN FILTOP := FILTOP + 1;
             WITH FILPTS[FILTOP] DO
             BEGIN  PTR := P;
               IF (NO = 8) ∧ (CL = 7) THEN ↱ IN ↳
                 SPEC := INP ELSE
               IF NO = 1 THEN
                IF AVAL = ΞOUTΞ THEN SPEC := OUT ELSE
                IF AVAL = ΞEXTΞ THEN SPEC := EXT ELSE
                IF AVAL = ΞPRINTΞ THEN SPEC := PRINT ELSE
                IF AVAL = ΞPUNCHΞ THEN SPEC := PUNCH
              ELSE FILERR    ELSE FILERR;
             END ↱WITH↳;
           END;
           INSYMBOL;  IF NO ≠ 12 THEN FILERR;
           INSYMBOL;
      END ↱NO = 11↳ ;
      ERR := FALSE;
      IF NO = 15 THEN
      BEGIN INSYMBOL;
         IF NO ≠ 1 THEN
         BEGIN ERROR(11);  GOTO 10 END;
      END ELSE IF NO ≠ 19 THEN ERROR(10);
   UNTIL NO ≠ 1 ;
   IF NO = 19 THEN INSYMBOL
      ELSE IF ¬ERR THEN ERROR(10) ;
   N := NEXT;  ERR := FALSE;
   TYPEDECL(TL,CTPTR);
   IF ERR THEN GOTO 10;
   LC := LC + I*TL;  LL := LC;  FPX := FILTOP;
   FOR I := I DOWNTO 1 DO
   WITH N↑ DO
   BEGIN LL := LL - TL;
      VTYPE := CTPTR;
      IF CTPTR ≠ NIL THEN
      IF CTPTR↑.FORM = CLASSS THEN
      BEGIN  ↱ CHECK FOR PREDECLARED CLASS ↳
         TYPID := NAME;  P := CTPTR↑.PELTYPE;
         IF PFTOP = FILLIMIT THEN ERROR(92) ELSE
         BEGIN PFTOP := PFTOP + 1;  PFL[PFTOP] := LL END;
         FOR J := PTX - 1 DOWNTO 0 DO
         WITH PTLIST[J] DO
```

```
                IF HNAME = TYPID THEN
                BEGIN PPTR↑.DOMAIN := N;
                   PPTR↑.ELTYPE := P;
                   PTX := PTX - 1;
                   HNAME := PTLIST[PTX].HNAME;
                   PPTR := PTLIST[PTX].PPTR;
                END  ↱ WITH, FOR ↳ ;
             VADDR := LL;
          END  ↱ CHECK CLASS ↳  ELSE
          IF CTPTR↑.FORM = FILES THEN
          BEGIN
            IF FILPTS[FPX].PTR ≠ N THEN ↱SCRATCH FILE↳
              IF FILTOP = FILLIMIT THEN ERROR(92) ELSE
              BEGIN FILTOP := FILTOP + 1;
                FILPTS[FILTOP].PTR := N;
                FILPTS[FILTOP].SPEC := SCRATCH;
              END ↱SCRATCH FILE↳
            ELSE FPX := FPX - 1;
            VADDR := LL;
            IF LEVEL > 0 THEN
            ↱ TAKE EFET INTO ACCOUNT FOR ADDRESS ↳
            BEGIN IF CTPTR↑.FELTYPE = CHARPTR THEN
                J := ALFALENG + 7 ELSE J := 7;
              LC := LC + J;  VADDR := VADDR + I*J - 7;
            END;
          END ↱FORM = FILES↳
          ELSE VADDR := LL ;
          N := NXTEL;
       END  ↱ FOR I ↳ ;
10 :   FINDSEMICOLON;
    END ↱WHILE NO = 1↳;
    IF NO = 47 THEN  ↱ VALUE ↳
       BEGIN IF LEVEL ≠ 0 THEN ERROR(22);
          INSYMBOL;
BEGIN IF LEVEL ≠ 0 THEN ERROR(22);
   INSYMBOL;
   WHILE NO = 1 DO
   BEGIN SRCHREC(NEXT);
      IF CTPTR = NIL THEN SEARCH;
      IF CTPTR = NIL THEN
      BEGIN ERROR(12);  FINDSEMICOLON;  GOTO 20 END;
      IF CTPTR↑.KLASS ≠ VARS THEN
      BEGIN ERROR(26);  FINDSEMICOLON;  GOTO 20 END;
      WITH CTPTR↑ DO
      BEGIN IT := VADDR;
         IF VLC = 0 THEN
         BEGIN VLC := IT;  LL := IT;  AT := IT;  END;
         IF IT < LL THEN ERROR(27) ELSE
         WHILE IT > AT DO
         BEGIN PUT(PASCLGO);  AT := AT + 1  END;
         INSYMBOL;
         IF (NO ≠ 8) ∨ (CL ≠ 6) THEN ERROR(4) ELSE INSYMBOL;
         IF NO = 9 THEN  ↱ LIST ↳
         BEGIN
            REPEAT INSYMBOL;  INCONST(I,PT,NEXT);  IT1 := 1;
               IF (NO = 6) ∧ (CL = 1) THEN   ↱  *  ↳
               BEGIN IT1 := I;  INSYMBOL;
                  INCONST(I,PT,NEXT);
               END;
               IF ¬ ERR THEN
              BEGIN
               FOR J := 1 TO IT1 DO
               BEGIN PASCLGO↑ := I;  PUT(PASCLGO)  END;
               AT := AT + IT1;
              END ↱¬ERR↳;
               IF ¬(NO IN [15,10]) THEN  ↱ NOT , NOR ) ↳
               BEGIN ERROR(20);  FINDSEMICOLON;  GOTO 20 END;
            UNTIL NO = 10 ;  INSYMBOL;
```

```
                  END ELSE
                  BEGIN INCONST(I,PT,NEXT);
                     IF ~ERR THEN
                     BEGIN PASCLGO↑ := I;  PUT(PASCLGO);
                        AT := AT + 1;
                     END;
                  END;
                  LL := IT + VTYPE↑.SIZE;
                  IF LL < AT THEN ERROR(20) ;
                  FINDSEMICOLON;
               END ↑WITH↓;
20:         END ↑WHILE NO = 1↓;
            FOR AT := AT + 1 TO LC DO PUT(PASCLGO) ;
         END ↑VALUE↓ ELSE VLC := LC;
   END  ↑ VARDECL ↓ ;

   PROCEDURE FORMPARM;
      ↑ ( HAS BEEN VERIFIED AND NEXT SYMBOL READ ↓
      LABEL 3;
      VAR SPEC : IDCLASS;  REP : BOOLEAN;

      PROCEDURE FORMERR;
      BEGIN ERROR(11);  SKIP(10);  GOTO EXIT 3 END;

   BEGIN SPEC := KONST ;
1:    REPEAT
         IF NO = 45 THEN ↑PROCEDURE↓
         REPEAT INSYMBOL;
            IF NO ≠ 1 THEN ERROR(11) ELSE
            BEGIN SRCHREC(NEXT);
               IF CTPTR ≠ NIL THEN ERROR(8) ELSE
               BEGIN NEW(P,PROC);
                  WITH P↑ DO
                  BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := PROC;
                    PROCTYPE := P;  PROCKIND := FORMAL;
                    PROCADDR := LC;  PROCLEVEL := LEVEL;
                    FORMALS := NIL; SEGSIZE := 0
                  END;  LC := LC + 1;  NEXT := P;
            END ↑CTPTR = NIL↓;
            INSYMBOL;
      END;
   UNTIL NO ≠ 15
   ELSE ↑NOT PROCEDURE↓
   BEGIN
      IF NO ≠ 1 THEN
      BEGIN IF NO = 43 THEN SPEC := VARS ELSE
            IF NO = 41 THEN
            BEGIN ERROR(24); SPEC := KONST END ELSE
            IF NO = 44 THEN SPEC := PROC ELSE FORMERR;
         INSYMBOL
      END ELSE SPEC := KONST;
      IF NO ≠ 1 THEN FORMERR;
      REPEAT
         SRCHREC(NEXT);
         IF CTPTR ≠ NIL THEN ERROR(3) ELSE
         BEGIN
            IF SPEC = PROC THEN
            BEGIN NEW(P,PROC);
              WITH P↑ DO
              BEGIN KLASS := PROC; PROCTYPE := NIL;
               PROCKIND := FORMAL; PROCADDR := LC;
               PROCLEVEL := LEVEL; FORMALS := NIL;
               SEGSIZE := 0
             END;
            END ELSE
            BEGIN NEW(P,VARS);
```

```
            WITH P↑ DO
            BEGIN KLASS := VARS; VTYPE := NIL;
              IF SPEC = VARS THEN VKIND := FORMAL
                             ELSE VKIND := ACTUAL;
              VADDR := LC; VLEVEL := LEVEL
            END
          END;
          P↑.NAME := AVAL;  P↑.NXTEL := NEXT;
          LC := LC + 1;  NEXT := P;
        END ↑CTPTR = NIL↓;
        INSYMBOL;
        IF NO = 15 THEN
        BEGIN INSYMBOL;
          IF NO = 19 THEN ERROR(11);
        END ELSE IF NO ≠ 19 THEN GOTO 4;
      UNTIL NO ≠ 1;
4:    IF NO ≠ 19 THEN ERROR(10) ELSE INSYMBOL;
      IF NO ≠ 1 THEN FORMERR;
      SEARCH;
      IF CTPTR = NIL THEN
      BEGIN ERROR(12);  GOTO 2 END;
      IF CTPTR↑.KLASS ≠ TYPES THEN
      BEGIN ERROR(18) ; GOTO 2 END ;
      N := NEXT;
      REPEAT
        IF SPEC = PROC THEN
          IF N↑.PROCTYPE = NIL THEN
            IF CTPTR↑.FORM > POWER THEN
            BEGIN ERROR(93); N := NIL END ELSE
            BEGIN N↑.PROCTYPE := CTPTR; N := N↑.NXTEL END
          ELSE N := NIL
        ELSE IF N↑.VTYPE = NIL THEN
              BEGIN N↑.VTYPE := CTPTR;
                IF N↑.VKIND = ACTUAL THEN
                  IF CTPTR↑.FORM IN [CLASSS,FILES] THEN ERROR(24);
                N := N↑.NXTEL
              END
            ELSE N := NIL
        UNTIL N = NIL;
2:      INSYMBOL;
      END ↑NOT PROCEDURE↓;
3:    REP := NO IN [1,41,43,44,45];
      IF NO = 16 THEN
      BEGIN INSYMBOL;  REP := NO ≠ 10  END;
    UNTIL ~REP;
    IF NO ≠ 10 THEN
    BEGIN ERROR(9);  SKIP(10);
      IF NO IN [41,43,44,45] THEN GOTO 1;
    END;
    ↑REVERSE POINTERS↓
    ↑AND RESERVE LOCAL CELLS FOR COPIES OF MULTIPLE VALUES↓
    N := NEXT;  NEXT := NIL;
    WHILE N ≠ NIL DO
    BEGIN P := N;
      WITH P↑ DO
      BEGIN N := NXTEL; NXTEL := NEXT;
        IF VTYPE ≠ NIL THEN
        IF (KLASS = VARS)∧(VKIND = ACTUAL)∧(VTYPE↑.SIZE ≠ 1) THEN
        BEGIN VADDR := LC;
          LC := LC + VTYPE↑.SIZE
        END;
      END;
      NEXT := P
    END;
  END ↑FORMPARM↓;
```

```
   PROCEDURE OUTPMD(SURRPTR : CTP);
      CONST DIRECT = 0, INDIRECT = 1;
      VAR N,P : CTP;  ADR,ACC : SHRTINT;  OUT : BOOLEAN;
   BEGIN
      WITH PMDFILE↑ DO
      BEGIN IF SURRPTR = NIL THEN
         BEGIN PNAME := '(MAIN)';  SA := IC END ELSE
         BEGIN PNAME := SURRPTR↑.NAME; SA := SURRPTR↑.PROCADDR END;
         PKIND := PROCDR;  LINK := LASTLINK;  COUNT := 0;
      END;
      PUT(PMDFILE);  PMCTR := PMCTR + 2;  LASTLINK := PMCTR;
      N := NEXT;
      WHILE N ≠ NIL DO
      WITH N↑ DO
      BEGIN OUT := FALSE;  ACC := DIRECT;
         IF KLASS = VARS THEN
         BEGIN OUT := TRUE; P := VTYPE; ADR := VADDR;
            IF VKIND = FORMAL THEN ACC := INDIRECT;
         END;
         IF OUT THEN IF P↑.FORM < POWER THEN
         BEGIN WITH PMDFILE↑ DO
            BEGIN PNAME := NAME;  PKIND := OBJ;
               IF P↑.FORM = NUMERIC THEN TYP := 1 ELSE
               IF P = REALPTR THEN TYP := 2 ELSE
               IF P = ALFAPTR THEN TYP := 3 ELSE
               IF P = CHARPTR THEN TYP := 4 ELSE
               IF P = BOOLPTR THEN TYP := 5 ELSE
               IF P↑.FORM = SYMBOLIC THEN TYP := 1 ELSE TYP := 6 ;
               RLA := ADR;  ACCESS := ACC;
            END;
            PUT(PMDFILE);  PMCTR := PMCTR + 2;
         END;
         N := NXTEL;
      END {WITH, WHILE};
   END {OUTPMD};

   PROCEDURE ENTERBODY ;
    {GENERATES  PROCEDURE ENTRY CODE, CODE
     TO INITIALIZE EFETS AND TO OPEN LOCAL FILES, AND CODE TO INITIALIZE
     LOCAL CLASSPOINTERS}
      CONST OPN = 104B ;
      VAR IT1,IT2 : SHRTINT; N : CTP;
   BEGIN CA := 0 ; CP := 4 ; LCX := 0 ;
      IF LEVEL ≠ 0 THEN
      BEGIN {PROCEDURE ENTRY CODE}
         GEN15(76B,0,5,0) ; GEN15(20B,0,0,18) ;
         GEN15(12B,7,7,0) ; GEN15(66B,5,6,0) ;
         GEN30(51B,7,5,1) ; GEN30(61B,6,6,0) ;
         LCA := CA ; LCP := CP ;
         IF LEVEL ≠ 1 THEN GEN15(56B,6,5,0) ;
         {CODE TO INITIALIZE EFETS}
         FOR IT := FILEV[LEVEL] TO FILTOP DO
         WITH FILPTS[IT].PTR↑ DO
         BEGIN
            UNPACK(NAME,LCH,1) ; IT2 := 7 ;
            WHILE LCH[IT2] = ' ' DO IT2 := IT2 - 1 ;
            RP := 0;  LDCST(ORD(NAME));
            GEN15(43B,0,0,6*IT2);  GEN15(11B,6,0,1);
            GEN30(51B,6,5,VADDR+2);
            GEN30(71B,6,0,VTYPE↑.FELTYPE↑.SIZE);
            IT1 := 2*(10B + SCRATCH {EOF AND SPEC} );
            IF VTYPE↑.FELTYPE ≠ CHARPTR THEN IT1 := IT1 + 1;
            IT1 := 128*IT1 + 30B ; {MASK}
            GEN30(71B,0,0,IT1);
            GEN15(20B,0,0,43);  GEN15(12B,6,6,0);
            GEN30(51B,6,5,VADDR+1);
```

```
      GEN30(71B,6,5,VADDR+7);
      GEN30(51B,6,5,VADDR+3);
      GEN30(72B,6,6,VTYPE↑.SIZE);
      GEN30(51B,6,5,VADDR+6);
   END;
END ELSE
BEGIN
   ↱MAIN PROGRAM, SAVE X7,B5 SET B6↳
   GEN15(56B,7,5,0) ; GEN15(76B,7,5,0) ;
   GEN30(51B,7,0,PARMLIST.IC0) ; GEN30(61B,6,5,0) ;
   LCA := CA ; LCP := CP ; LC := 1 ;
END ;
IF STOFLCHECK THEN GEN30(06B,6,4,STOFLERR) ;
↱COPY MULTIPLE VALUES INTO LOCAL CELLS↳
IF SURRPTR ≠ NIL THEN
BEGIN N := SURRPTR↑.FORMALS; IT := 2;
   WHILE N ≠ NIL DO
   BEGIN IT := IT + 1;
      WITH N↑ DO
      BEGIN IF VTYPE ≠ NIL THEN
         IF (KLASS = VARS)∧(VKIND = ACTUAL)∧(VTYPE↑.SIZE ≠ 1) THEN
         BEGIN GEN30(51B,1,5,IT); GEN30(71B,2,5,VADDR);
            GEN30(61B,7,0,VTYPE↑.SIZE-1); NOOP; IT2 := IC;
            GEN15(53B,3,1,7); GEN15(10B,6,3,0);
            GEN15(53B,6,2,7); GEN30(61B,7,7,-1);
           GEN30(06B,7,0,IT2)
         END;
         N := NXTEL
      END
   END
END;
↱CODE TO OPEN FILES↳
IF LEVEL ≠ 0 THEN IT2 := 5 ELSE
BEGIN IT2 := 0;  GEN30(61B,7,0,INPT+1);
  JUMPTO(GETC);  ↱GET(INPUT)↳
   END;
   FOR IT := FILEV[LEVEL] TO FILTOP DO
   BEGIN IT1 := FILPTS[IT].PTR↑.VADDR + 1;
     GEN30(61B,7,IT2,IT1);  JUMPTO(OPN);
     IF FILPTS[IT].SPEC IN [INP,EXT] THEN
     BEGIN GEN30(61B,7,IT2,IT1);
       IF FILPTS[IT].PTR↑.VTYPE↑.FELTYPE = CHARPTR THEN
         JUMPTO(GETC)  ELSE  JUMPTO(GETB) ;
     END;
   END;
   ↱CODE TO INITIALIZE CLASSPOINTERS↳
   FOR IT := PILEV[LEVEL] TO PFTOP DO
   BEGIN IT1 := PFL[IT] ;
      GEN30(71B,7,IT2,IT1+1) ; GEN30(51B,7,IT2,IT1) ;
   END ;
   PFTOP := PILEV[LEVEL] - 1 ; TCT := 0 ; TMAX := 0 ; CLABIX := 0 ;
END ↱ENTERBODY↳ ;

PROCEDURE LEAVEBODY ;
 ↱CALLED AT PROCEDURE END. GOES THROUGH LABELTABLE LABTAB AND
  EXITTABLE EXTAB, UPDATING CODE, AND GENERATES PROCEDURE EXIT CODE↳
   CONST CLS = 105B ;
   VAR IT2,IT3 : INTEGER;
   PROCEDURE ERRMESSAGE(ALF:ALFA; VAL:SHRTINT) ;
   BEGIN ERROR(43) ; WRITE(EOL) ; PRTERR ;
      WRITE(Ξ Ξ,ALF:6,VAL:6,EOL) ;
      IF ¬EOLFLAG THEN WRITE((Ξ Ξ):CHCNT+8)
   END ↱ERRMESSAGE↳ ;
BEGIN IF LEVEL = 0 THEN IT2 := 0 ELSE IT2 := 5 ;
   FOR IT := FILEV[LEVEL] TO FILTOP DO
   BEGIN GEN30(61B,7,IT2,FILPTS[IT].PTR↑.VADDR+1); JUMPTO(CLS);
```

```
END ;
FILTOP := FILEV[LEVEL] - 1 ;
IF LEVEL # 0 THEN ↑PROCEDURE EXIT CODE↓
BEGIN GEN30(51B,1,5,1) ; GEN15(66B,6,5,0) ;
   GEN15(63B,7,1,0) ; GEN15(20B,1,0,42) ;
   GEN15(63B,5,1,0) ;
END ELSE ↑MAIN, RETURN TO MONITOR↓
BEGIN GEN15(56B,1,5,0) ; GEN15(63B,7,1,0) END ;
GEN30(62B,7,0,0) ; LC := LC + TMAX ; INS(LC,LCP,LCA) ;
FOR IT := 1 TO CLABIX DO
WITH LABTAB[IT] DO
BEGIN IF FLD2 > 0 THEN ↑UNDECLARED  LABEL↓
   BEGIN
      ERRMESSAGE(≡LABEL:≡,LABVAL) ;
      UNDLAB[FLD2].SUCC := CHNIX ; CHNIX := FLD3 ;
   END ↑IF FLD2 > 0↓ ;
END ↑WITH, FOR↓ ;
FOR IT := FSTIX TO CEXTABIX DO
WITH EXTAB[IT] DO
BEGIN FOR IT1 := 1 TO CLABIX DO
   WITH LABTAB[IT1] DO
   IF LABVAL = EXVAL THEN
   BEGIN IF FLD2 > 0 THEN ERRMESSAGE(≡EXIT:≡,EXVAL) ELSE
      BEGIN IT2 := JMPCST ; INSERT(LC,30,IT2) ;
         INSERT(FLD3,0,IT2) ; JMPTAB[JMPTABIX] := IT2
      END ;
      GOTO 2
   END ↑IF, WITH, FOR↓ ;
   ERRMESSAGE(≡EXIT:≡,EXVAL) ;
```

```
:    END ↑WITH, FOR↓ ;
```

```
      CEXTABIX := FSTIX - 1 ;
      WRITOUT ;
      IF LEVEL = 0 THEN  ↑ WRITE OUT JUMPTABLE ↓
      FOR IT := 0 TO JMPMAX DO
      BEGIN PASCLGO↑ := JMPTAB[IT] ; PUT(PASCLGO) END
   END ↑LEAVEBODY↓ ;

   BEGIN  ↑BODY↓
      PILEV[LEVEL]:= PFTOP+1;  VLC := 0;
      FILEV[LEVEL] := FILTOP + 1;
      FSTIX := CEXTABIX + 1 ;
```

```
{     IF NO = 40 THEN #LABEL#
      BEGIN REPEAT INSYMBOL ;
            IF (NO = 2)^(CL = 1) THEN
            BEGIN IF IVAL ≥ TWOTO17 THEN
               BEGIN ERROR(100) ; GOTO 2 END ;
               FOR IT := FSTIX TO CEXTABIX DO
               IF EXTAB[IT].EXVAL = IVAL THEN
               BEGIN ERROR(77) ; GOTO 2 END ;
               IF CEXTABIX = MAXEXLABS THEN ERROR(78) ELSE
               BEGIN CEXTABIX := CEXTABIX + 1 ;
                  IF JMPIX > JMPMAX THEN ERROR(91) ELSE
                  BEGIN WITH EXTAB[CEXTABIX] DO
                     BEGIN EXVAL := IVAL ; JMPTABIX := JMPIX END ;
                     JMPTAB[JMPIX] := LEVEL ; JMPIX := JMPIX + 1 ;
                  END
               END ;
```

```
      END ↑IF (NO=2)∧(CL=1)↓ ELSE
      BEGIN ERROR(61) ; GOTO 3 END
   UNTIL NO ≠ 15 ↑,↓ ;




:     FINDSEMICOLON;
   END ;
   IF NO = 41 THEN ↑ CONST ↓
   BEGIN INSYMBOL ;
      WHILE NO = 1 DO
      BEGIN REPEAT SRCHREC(NEXT) ; IF CTPTR ≠ NIL THEN ERROR(8);
         NEW(P,KONST);
         WITH P↑ DO
         BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := KONST;
         END;  NEXT := P;
         INSYMBOL;
         IF (NO ≠ 8) ∨ (CL ≠ 6) THEN ERROR(4) ELSE INSYMBOL;
         IF NO = 36 THEN  ↑ NIL ↓
         BEGIN P↑.CONTYPE := NILPTR;
            P↑.VALUES := NILVAL;  INSYMBOL;
         END ELSE
         BEGIN INCONST(I,N,NEXT↑.NXTEL);
            P↑.CONTYPE := N;  P↑.VALUES := I;
         END;
            WHILE NO = 15 DO
            BEGIN INSYMBOL;
               IF NO ≠ 1 THEN
               BEGIN ERROR(11); SKIP(15) END;
            END;
         UNTIL NO ≠ 1;
            END;
         UNTIL NO ≠ 1;
         FINDSEMICOLON;
      END ↑ WHILE NO = 1 ↓ ;
   END ↑CONST↓;
   IF NO =37 THEN  ↑TYPE↓
   BEGIN INSYMBOL;
   WHILE NO = 1 DO
   BEGIN SRCHREC(NEXT); IF CTPTR ≠ NIL THEN ERROR(8);
      TYPID := AVAL;  INSYMBOL;
      IF (NO ≠ 3) ∨ (CL ≠ 6) THEN ERROR(4) ELSE INSYMBOL;
      ERR := FALSE;  TYPEDECL(TL,P);
      IF ¬ERR THEN
      IF P↑.NAME ≠ BLANK THEN ERROR(96) ELSE
      BEGIN P↑.NAME := TYPID;  P↑.NXTEL := NEXT;
         NEXT := P;
      END;
      FINDSEMICOLON;
   END;
   END  ↑TYPE↓ ;
   IF NO=43 THEN  ↑VAR↓
      VARDECL ELSE VLC := LC;
   IF PTX > 0 THEN
   BEGIN ERROR(12); WRITE(EOL); PRTERR;
      FOR PTX := PTX-1 DOWNTO 0 DO
         WRITE(Ξ Ξ,ΞCLASS-ID  Ξ,PTLIST[PTX].HNAME,EOL) ;
      IF ¬EOLFLAG THEN WRITE((Ξ Ξ):CHCNT+3);
   END;
   IF LEVEL = 0 THEN
   BEGIN IF ¬PARMLIST.ERRFLAG THEN
      FOR I := 0 TO FILTOP DO
      WITH FILPTS[I].PTR↑ DO
      BEGIN
         IF VTYPE↑.FELTYPE = CHARPTR THEN
         BEGIN IT := 0;  LC := LC + ALFALENG;
```

```
            FOR J := 1 TO ALFALENG DO PUT(PASCLGO);  ↑CHAR BUFFER↓
         END ELSE IT := 1;  ↑ C/B BIT ↓
         PUT(PASCLGO);   ↑ P-PTR ↓
         J := FILPTS[I].SPEC;
         IF ¬(J IN [INP,EXT]) THEN J := J + 8; ↑EOF BIT↓
         J := (2*J + IT)*128 + 30B ; ↑APPEND C/B BIT AND MASK↓
         UNPACK(NAME,LCH,1) ; IT1 := 7 ;
         WHILE LCH[IT1] = ≡ ≡ DO IT1 := IT1 - 1 ;
         IT1 := (ALFALENG - IT1)*6 ;
         APPEND(J,48,VTYPE↑.FELTYPE↑.SIZE);
         PASCLGO↑ := J;  PUT(PASCLGO);  ↑EOF,SPEC,C/B,LRL↓
         J := ORD(NAME);  APPEND(J,-IT1,0);  APPEND(J,IT1,0);
         PASCLGO↑ := J;  PUT(PASCLGO);  ↑ LFN ↓
         PASCLGO↑ := VADDR;
         FOR J := 1 TO 3 DO PUT(PASCLGO);  ↑FIRST,IN,OUT↓
         PASCLGO↑ := VADDR + VTYPE↑.SIZE;  PUT(PASCLGO);   ↑ LIMIT ↓
         VADDR := LC;  LC := LC + 7;
      END;
      PARMLIST.LPJMPTAB := LC ;
      IF VLC ≠ LC THEN FOR J := 0 TO JMPMAX DO PUT(PASCLGO)
                   ELSE VLC := LC + (JMPMAX + 1) ;
      LC := LC + (JMPMAX + 1);
      IC := LC;  PARMLIST.LOADPT := VLC;  ↑LOADPT↓
   END;
   DP := FALSE;
   IF NO IN [44,45] THEN ↑FUNCTION OR PROCEDURE↓
   BEGIN IF SURRPTR ≠ NIL THEN
         IF SURRPTR↑.PROCADDR = 0 THEN
         BEGIN IF JMPIX > JMPMAX THEN ERROR(91) ELSE
            BEGIN SURRPTR↑.PROCADDR := PARMLIST.LPJMPTAB + JMPIX ;
               JMPIX := JMPIX + 1 ;
            END
         END;
      REPEAT
         LL := NO; ERR := FALSE;
         OLDLEV := LEVEL;
         IF LEVEL < MAXLEVEL THEN LEVEL := LEVEL + 1 ELSE ERROR(76);
         INSYMBOL;
         IF NO ≠ 1 THEN
         BEGIN ERROR(11);  LEVEL := OLDLEV;  GOTO 1 END;
         LC1 := LC;  SRCHREC(NEXT);
         IF CTPTR ≠ NIL THEN
         IF CTPTR↑.KLASS ≠ PROC THEN
         BEGIN ERROR(8); CTPTR := NIL END;
         IF CTPTR = NIL THEN  ↑ UNDECLARED PROCEDURE ↓
         BEGIN NEW(PROCPTR,PROC);
            WITH PROCPTR↑ DO
            BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := PROC;
               PROCTYPE := PROCPTR;  PROCKIND := ACTUAL;
               PROCLEVEL := LEVEL - 1;
            END;
            DISPLAY[TOP].FNAME := PROCPTR;
            NEXT := NIL ;
            INSYMBOL;
            IF LL = 44 THEN  ↑ FUNCTION ↓
            BEGIN IF NO ≠ 9 THEN ERROR(29) ELSE
               BEGIN LC := 3 ; INSYMBOL ; FORMPARM END ;
               IF ERR THEN PROCPTR↑.PROCTYPE := NIL;
               IF NO = 10 THEN
               BEGIN INSYMBOL;
                 IF NO ≠ 19 THEN ERROR(10) ELSE INSYMBOL;
                 IF NO ≠ 1 THEN
                 BEGIN ERROR(11);  SKIP(49);
                   PROCPTR↑.PROCTYPE := NIL;
                 END ELSE
                 BEGIN SEARCH;
```

```
      IF CTPTR ≠ NIL THEN
      BEGIN IF CTPTR↑.KLASS ≠ TYPES THEN
         BEGIN ERROR(93); CTPTR := NIL END ELSE
         IF CTPTR↑.FORM > POWER THEN
         BEGIN ERROR(93) ; CTPTR := NIL END ;
      END ELSE ERROR(12) ;
      PROCPTR↑.PROCTYPE := CTPTR ;
      INSYMBOL;
        END;
      END ↱NO = 10↳;
   END  ↱ FUNCTION ↳  ELSE  ↱ PROCEDURE ↳
   BEGIN LC := 3 ;
      IF NO = 9 THEN  ↱ PARMLIST ↳
      BEGIN INSYMBOL; FORMPARM;
         IF ERR THEN PROCPTR↑.PROCTYPE := NIL;
         IF NO = 10 THEN INSYMBOL;
      END ↱NO = 9↳;
   END ↱PROCEDURE↳;
   IF NO ≠ 16 THEN
   BEGIN PROCPTR↑.PROCTYPE := NIL;
      ERROR(58);  SKIP(16);
   END ELSE INSYMBOL;
   PROCPTR↑.FORMALS := NEXT;
   IF (NO=1) ∧ (AVAL=≡FORWARD≡) THEN
   BEGIN NEXT := PROCPTR;
      IF JMPIX > JMPMAX THEN ERROR(91) ELSE
         BEGIN PROCPTR↑.PROCADDR :=
            PARMLIST.LPJMPTAB + JMPIX ;
            JMPIX := JMPIX + 1 ;
         END ;
      PROCPTR↑.SEGSIZE := -LC;
       ↱ SEGSIZE < 0 SIGNIFIES FORWARD-DECLARATION ↳
      INSYMBOL;
   END ELSE
   BEGIN TOP := LEVEL + 1;
      WITH DISPLAY[TOP] DO
      BEGIN FNAME := NEXT;  OCCUR := BLOCK  END;
      PROCPTR↑.PROCADDR := 0;
      NEW(LFIRSTENTRY,DUMMYCLASS) ;
      BODY(PROCPTR,LFIRSTENTRY);
   END;
END  ↱ NEW PROC ↳  ELSE  ↱ PROC ID ALREADY DECLARED ↳
BEGIN
   IF CTPTR↑.SEGSIZE ≥ 0 THEN  ↱ PREV. DECL NOT FORWARD ↳
      ERROR(16);
   INSYMBOL;
   IF NO = 9 THEN  ↱ IGNORE PARM-LIST ↳
   BEGIN ERROR(23);
      REPEAT SKIP(10);
         IF NO IN [16,41,43,44,45] THEN INSYMBOL;
      UNTIL ¬(NO IN [16,41,43,44,45]);
      IF NO ≠ 10 THEN ERROR(9) ELSE INSYMBOL;
   END;
   IF NO = 15 THEN SKIP(16);
   IF NO ≠ 16 THEN ERROR(58) ELSE INSYMBOL;
   IF (NO=1) ∧ (AVAL=≡FORWARD≡) THEN ↱AGAIN FORWARD↳
      INSYMBOL ELSE
  BEGIN
   PROCPTR := CTPTR;
   WITH PROCPTR↑ DO
   BEGIN LC := -SEGSIZE;  NEXT := FORMALS  END;
   TOP := LEVEL + 1;
   WITH DISPLAY[TOP] DO
   BEGIN FNAME := NEXT;  OCCUR := BLOCK  END;
   NEW(LFIRSTENTRY,DUMMYCLASS);
   BODY(PROCPTR,LFIRSTENTRY);
```

```
      END ↑NOT FORWARD↓;
    END  ↑ OLD PROCEDURE ↓ ;
    LC := LC1;
    LEVEL := OLDLEV;
    FINDSEMICOLON;
  UNTIL ¬(NO IN [44,45]) ;
END  ↑ FUNCTION OR PROCEDURE ↓ ;
DISPLAY[TOP].FNAME := NEXT;
IF NO≠21 THEN
BEGIN ERROR(24); SKIP(49);
  WHILE NO IN [16,22] DO
  BEGIN INSYMBOL;
    SKIP(49);
  END ;
  IF NO IN [37,40,41,43,44,45] THEN GOTO 1;
END ELSE
IF SURRPTR = NIL THEN PARMLIST.ENTRYPT := IC ELSE
WITH SURRPTR↑ DO
BEGIN IF PROCADDR ≠ 0 THEN
  BEGIN I := IC ; APPEND(I,30,EQJMP) ;
    JMPTAB[PROCADDR - PARMLIST.LPJMPTAB] := I
  END ;
  PROCADDR := IC
END ;
N := NEXT ; ERR := FALSE ;
WHILE N ≠ NIL DO
  IF N↑.KLASS ≠ PROC THEN N := NIL ELSE
  BEGIN IF N↑.SEGSIZE < 0 THEN ↑FORWARD, NO REAL DECLARATION↓
    BEGIN IF ¬ERR THEN
      BEGIN ERROR(12) ; WRITE(EOL) ; PRTERR END ;
      WRITE(Ξ PROCEDUREΞ,Ξ  Ξ,N↑.NAME,EOL) ;
    END ;
    N := N↑.NXTEL ;
  END ↑IF, WHILE↓ ;
  IF ERR THEN IF ¬EOLFLAG THEN WRITE((Ξ Ξ):CHCNT+8) ;
  IF ¬PARMLIST.ERRFLAG THEN OUTPMD(SURRPTR) ;
  ENTERBODY;
  COMPSTAT ;
  LEAVEBODY ;
  IF SURRPTR ≠ NIL THEN SURRPTR↑.SEGSIZE := LC;
  NEW(P,DUMMYCLASS);
  IF P > MAXCTP THEN MAXCTP := P;
  RESET(FIRSTENTRY);
  TOP := LEVEL;  NEXT := DISPLAY[TOP].FNAME;
END  ↑ BODY ↓ ;

↑ ------------------------------------------------------------ ↓
↑  M A I N   P R O G R A M  ↓

BEGIN

  ↑ INITIALIZE CONTEXT-TABLE ↓
  NEW(NILPTR,TYPES,POINTER);   ↑TYPE OF NIL↓
  WITH NILPTR↑ DO
  BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
    SIZE := 0;  FORM := POINTER;  DOMAIN := NIL;  ELTYPE := NIL;
  END;
  NEXT := NIL ;
  FOR IT := 1 TO 12 DO    ↑WEOR, GET, PUT, RESET, REWRITE, PACK↓
  BEGIN NEW(PT,PROC);   ↑UNPACK, INSERT, APPEND, READ, WRITE, ALLOC↓
    WITH PT↑ DO
    BEGIN NAME := INITNAM[IT-2];  NXTEL := NEXT;  KLASS := PROC;
      PROCTYPE := PT;  FORMALS := NIL;
      PROCADDR := 0 ; PROCLEVEL := 0 ; SEGSIZE := IT ;
      PROCKIND := ACTUAL;
    END;  NEXT := PT;
```

```
   END;
   FOR IT := 1 TO 9 DO
   BEGIN NEW(PT,PROC);   (*ODD, INT, CHR, EOF, ABS, SQR, TRUNC,*)
      WITH PT^ DO          (*PRED, SUCC*)
      BEGIN NAME := INITNAM[IT+10]; NXTEL := NEXT; KLASS := PROC;
         PROCTYPE := NILPTR; FORMALS := NIL;
         PROCADDR := 0 ; PROCLEVEL := 0 ; SEGSIZE := IT ;
         PROCKIND := ACTUAL;
      END; NEXT := PT;
   END;
   NEW(CTPTR,TYPES,FILES);   (*TEXT*)
   WITH CTPTR^ DO
   BEGIN NAME := ΞTEXTΞ; NXTEL := NEXT; KLASS := TYPES;
      SIZE := 513; FORM := FILES;
   END; FILPTS[1].PTR := CTPTR;
   NEXT := CTPTR ;
   FOR IT := 22 TO 23 DO
   BEGIN NEW(PT,VARS);   (*INPUT, OUTPUT*)
      WITH PT^ DO
      BEGIN NAME := INITNAM[IT]; NXTEL := NEXT; KLASS := VARS;
         VTYPE := CTPTR; VKIND := ACTUAL;
         IF IT = 22 THEN
            VADDR := INPT ELSE VADDR := OUTPT ;
         VLEVEL := 0;
      END; NEXT := PT;
   END;
   NEW(ALFAPTR,TYPES,SYMBOLIC);   (*ALFA*)
   WITH ALFAPTR^ DO
   BEGIN NAME := INITNAM[24]; NXTEL := NEXT; KLASS := TYPES;
      SIZE := 1; FORM := SYMBOLIC; FCONST := NIL; PWSET := NIL;
      BITSIZE := WORDLENGTH;
   END;
   NEW(REALPTR,TYPES,SYMBOLIC);   (*REAL*)
   WITH REALPTR^ DO
   BEGIN NAME := INITNAM[25]; NXTEL := ALFAPTR; KLASS := TYPES;
      SIZE := 1; FORM := SYMBOLIC; FCONST := NIL; PWSET := NIL;
      BITSIZE := WORDLENGTH;
   END;
   NEW(CHARPTR,TYPES,SYMBOLIC);   (*CHAR*)
   WITH CHARPTR^ DO
   BEGIN NAME := INITNAM[26]; NXTEL := REALPTR; KLASS := TYPES;
      SIZE := 1; FORM := SYMBOLIC; BITSIZE := 6;
   END;
   NEW(PT,KONST);   (*Ξ;Ξ*)
   WITH PT^ DO
   BEGIN NAME := BLANK; NXTEL := NIL; KLASS := KONST;
      CONTYPE := CHARPTR;
      VALUES := 63; SUCC := NIL;
   END; CHARPTR^.FCONST := PT;
   NEW(BOOLPTR,TYPES,SYMBOLIC);   (*BOOLEAN*)
   WITH BOOLPTR^ DO
   BEGIN NAME := INITNAM[27]; NXTEL := CHARPTR; KLASS := TYPES;
      SIZE := 1; FORM := SYMBOLIC; BITSIZE := 1;
   END;
   NEW(PT,KONST);   (*EOL*)
   WITH PT^ DO
   BEGIN NAME := INITNAM[20]; NXTEL := BOOLPTR;
      KLASS := KONST; CONTYPE := CHARPTR;
      VALUES := 0; SUCC := NIL
   END ;
   NEXT := PT; CTPTR := NIL;
   FOR IT := 0 TO 1 DO
   BEGIN NEW(PT,KONST);   (*FALSE, TRUE*)
      WITH PT^ DO
      BEGIN NAME := INITNAM[IT+28]; NXTEL := NEXT;
         KLASS := KONST; CONTYPE := BOOLPTR;
```

```
      VALUES := IT;  SUCC := CTPTR;
   END;  CTPTR := PT;  NEXT := PT;
END;
BOOLPTR↑.FCONST := NEXT;
NEW(INTPTR,TYPES,NUMERIC);   ↗INTEGER↘
WITH INTPTR↑ DO
BEGIN NAME := INITNAM[30];  NXTEL := NEXT;  KLASS := TYPES;
   SIZE := 1 ; FORM := NUMERIC ;
   MIN := 777740000000000000B;  MAX := 37777777777777778;
   BITS := WORDLENGTH;
END;  PREDEFP := INTPTR;
FOR IT := 1 TO 2 DO
BEGIN NEW(PT,TYPES,POWER);   ↗POWERSET CHAR, POWERSET BOOLEAN↘
   WITH PT↑ DO
   BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
      SIZE := 1;  FORM := POWER;
      IF IT = 1 THEN ↗POWERSET BOOLEAN↘
      BEGIN ELSET := BOOLPTR;  BOOLPTR↑.PWSET := PT;  PWBITS := 3;
      END ELSE ↗POWERSET CHAR↘
      BEGIN ELSET := CHARPTR;  CHARPTR↑.PWSET := PT;
         PWBITS := WORDLENGTH;
      END;
   END;
END;
NEW(UNDECPTR,VARS);   ↗TYPE OF ALL UNDECLARED IDES↘
WITH UNDECPTR↑ DO
BEGIN NAME := BLANK ; NXTEL := NIL ; KLASS := VARS ; VTYPE := NIL ;
   VKIND := ACTUAL ; VADDR := 0 ; VLEVEL := 0 ;
END ;
NEW(PNUMPTR,TYPES,POWER);   ↗TYPE OF POWERSET 0..58↘
WITH PNUMPTR↑ DO
BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
   SIZE := 1;  FORM := POWER;  ELSET := INTPTR;  PWBITS := WORDLENGTH;
END;
NEW(LAMPTR,TYPES,POWER);   ↗TYPE OF [ ]↘
WITH LAMPTR↑ DO
BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
   SIZE := 1;  FORM := POWER;  ELSET := NIL;  PWBITS := 0;
END;
NEW(CTPTR,VARS);   ↗TYPE OF PARAM OF EXTERNAL PROCEDURES↘
WITH CTPTR↑ DO
BEGIN NAME := BLANK ; NXTEL := NIL ; KLASS := VARS ;
   VTYPE := REALPTR; VKIND := ACTUAL;
   VADDR := 3; VLEVEL := 0;
END ;
NEXT := INTPTR ;
NEW(PT,KONST);   ↗ALFALENG↘
WITH PT↑ DO
BEGIN NAME := INITNAM[21] ; NXTEL := NEXT ; KLASS := KONST ;
   CONTYPE := INTPTR ; VALUES := 10 ;
   SUCC := NIL
END ;
NEXT := PT ;
FOR IT := 32 TO 37 DO
BEGIN NEW(PT,PROC);   ↗SIN, COS, EXP, LN, SQRT, ARCTAN↘
   WITH PT↑ DO
   BEGIN NAME := INITNAM[IT] ; NXTEL := NEXT ;
      KLASS := PROC ; PROCTYPE := REALPTR ;
      PROCKIND := ACTUAL ; PROCLEVEL := 0 ;
      FORMALS := CTPTR; PROCADDR := IT + 58;
      IF IT ≤ 33 THEN SEGSIZE := 0 ELSE SEGSIZE := IT - 33 ;
   END ;
   NEXT := PT
END ;
EXTPTR := NEXT ;
FILPTS[1].PTR↑.FELTYPE := CHARPTR;
```

```
    DIGITS := [ΞOΞ,Ξ1Ξ,Ξ2Ξ,Ξ3Ξ,Ξ4Ξ,Ξ5Ξ,Ξ6Ξ,Ξ7Ξ,Ξ8Ξ,Ξ9Ξ];
    FOR IT := 0 TO 3 DO PARMLIST.ERRNRS[IT] := [ ] ;
    ERRINX := 0;  EOLFLAG := FALSE;  CHCNT := 1;
    PARMLIST.ERRFLAG := FALSE ;
    DP := TRUE;  PRCODE := FALSE;
    PMCTR := 0;  LASTLINK := 0;  (*FOR PMDFILE*)
    ASSCHECK := TRUE ; INXCHECK := TRUE ; DIVCHECK := TRUE ;
    STOFLCHECK := TRUE ; ROUNDING := TRUE ;
    KK := ALFALENG; FILTOP := -1;  JMPIX := 0;  CEXTABIX := 0;
    PTX := 0;  CHNIX := 1;  B6DPL := 3;
    FOR IT := 1 TO UNDMAX-1 DO UNDLAB[IT].SUCC := IT + 1 ;
    UNDLAB[UNDMAX].SUCC := 0 ;
    IC := PARMLIST.IC0 + 1 ;
    MAXCTP := LAMPTR;
    PARMLIST.EXTFLAGS := [ ] ;
    WITH DISPLAY[0] DO
    BEGIN FNAME := EXTPTR ; OCCUR := BLOCK END ;
    NO := 0; LC := IC;
    CH := INPUT↑;  WRITE(Ξ Ξ,LC : 6 OCT,Ξ Ξ,CH);
    INSYMBOL;
    REPEAT
       DISPLAY[1].FNAME := NIL;  DISPLAY[1].OCCUR := BLOCK;
       TOP := 1;  LEVEL := 0;  NEXT := NIL;
       NEW(PT,DUMMYCLASS); LC := IC ; PFTOP := -1;

      (*********************
       *                   *
       *↓ BODY(NIL,PT); (*  COMPILE USER PROGRAM
       *                   *
       *********************↓

       IF (LEVEL = 0)^(NO ≠ 17) THEN ERROR(54) ;
      UNTIL NO = 17 (* . ↓ ;
      IF ¬EOLFLAG THEN
      REPEAT GET(INPUT);  WRITE(INPUT↑) UNTIL INPUT↑ = EOL;
      IF ERRINX > 0 THEN PRTERR;
      IF PRCODE THEN PRJMPTAB;
      PMDFILE↑.PKIND := PROCDR; PMDFILE↑.COUNT := LASTLINK;
      PUT(PMDFILE);  (*LINK TO MAIN PROGRAM ENTRY↓
      WRITE(EOL) ;
      PARMLIST.LIMCODE := IC ;
    END .
```

```
            IDENT    SINCOS
            ENTRY    SIN
            ENTRY    COS
B1B2        BSS      1                                        PASCAL
SIN         SB7      0                                        PASCAL
            EQ       SINCOS                                   PASCAL
COS         SB7      1                                        PASCAL
SINCOS      SA7      B6+1            STORE RETURN ADDRESS     PASCAL
            MX1      42                                       PASCAL
            SX6      B1                                       PASCAL
            BX6      -X1*X6                                   PASCAL
            LX6      18                                       PASCAL
            SX0      B2                                       PASCAL
            BX0      -X1*X0                                   PASCAL
            BX6      X6+X0                                    PASCAL
            SA6      B1B2                                     PASCAL
                                                              PASCAL
                                                              PASCAL
            SA3      B6+3            GET ARGUMENT             PASCAL
            SA2      TWOPI                                    PASCAL
            OR       X3,ERR2
            ID       X3,ERR3
            FX7      X3*X2           .N = ARG*2/PI
```

```
          MX0       0
          SB1       1
          PX2       X0              .GET MASK = 2000000000000000000B
          FX4       X2+X7           .
          DX6       X2+X7           .
          RX0       X4+X6           .ROUNDED N IN X0
          NX1       B2,X0           .NORMALIZE N FOR RANGE REDUCTION
          BX6       X2-X0           .N IN INTEGER FORMAT
          IX0       X2+X6           .MAKE INTEGER POSITIVE
          SA2       A2+B1           .PI/2-U
          EQ        B2,B0,ERR1      .JUMP IF ARG TOO LARGE
                                                                    PASCAL
          FX7       X1*X2           .N*(PI/2-U)U = H1
          SA4       A2+B1           .PI/2-L
          ZR        X6,NRR          .N=0, NO RANGE REDUCTION NECESSAR
          FX6       X3-X7           .(ARG-H1)U
          DX5       X2*X1           .N*(PI/2-U)L = H2
          DX2       X3-X7           .(ARG-H1)L
          NX6       B0,X6           .
          FX7       X4*X1           .N*(PI/2-L)U
          FX2       X2-X5           .
          SX1       B7              .K TO X1
          FX2       X2-X7           .LOW ORDER TERMS
          IX0       X0+X1           .N + K + 2**58,,, .GT. 0
          MX5       58              .X5=-3
          BX7       -X5*X0          .  K = (N+K)MOD4
          RX6       X6+X2           .FINAL REDUCED ARG= X
          SB7       X7              .K BACK TO B7                   PASCAL
          NX3       B0,X6           .   NORMALIZED
NRR       SX6       B7              .K TO X6                        PASCAL
          FX0       X3*X3           .Z=X*X
          LX6       59              .K = 0,2 USE SINE EVALUATION
          PL        X6,SINE         .
          SA1       A4+B1           .C5
          FX7       X0*X0           .Z**2
          SA2       A1+B1           .C6
          FX4       X0*X1           .C5*Z
          FX5       X2*X7           .C6*Z**2
          SA3       A2+B1           .C4
          FX6       X4+X3           .C5*Z+C4
          SA4       A3+B1           .C3
          FX1       X4*X0           .C3*Z
          FX6       X6+X5           .C6*Z*Z+C5*Z+C4 = ( 1 )
          FX2       X6*X7           .( 1 )*Z**2
          SA3       A4+B1           .C2
          FX6       X1+X3           .C3*Z + C2
          SA4       A3+B1           .C1
          FX6       X2+X6           .( 1 )Z**2+C3*Z+C2 = ( 2 )
          FX0       X0*X4           .C1*Z
          SA5       A4+B1           .C0
          FX7       X6*X7           .( 2 )*Z**2
          FX6       X7+X0           .( 2 )Z**2 + C1*Z
          RX0       X6+X5           .C3 + REST OF SERIES
          JP        CONT            .DONE WITH COSINE SERIES
SINE      SA1       S5              .S5
          FX6       X1*X0           .S5*Z
          SA2       A1+B1           .S4
          FX7       X0*X0           .Z**2
          FX5       X2+X6           .S5*Z+S4
          SA1       A2+B1           .S3
          SA2       A1+B1           .S2
          FX6       X5*X7           .(S5*Z+S4)*Z**2
          FX4       X1*X0           .S3*Z
          FX5       X6+X2           .(S5*Z+S4)Z**2 + S2
          SA2       A2+B1           .S1
          FX5       X5+X4           .(S5*Z+S4)Z**2+S2+S3*Z = ( 3)
```

```
          FX7     X5*X7           .( 3 )*Z**2
          FX3     X2*X0           .S1*Z
          SA1     A2+B1           .S0
          FX4     X0+X7           .
          RX5     X4+X1           .
          RX6     X3*X5           .FINAL SINE TERM
CONT      SX3     B1-B7           .1-K                                    PASCAL
          AX3     2               .SIGN(1-K)                              PASCAL
          BX6     X0-X3                                                   PASCAL
EXIT      SA1     B1B2                                                    PASCAL
          SB2     X1                                                      PASCAL
          AX1     18                                                      PASCAL
          SB1     X1                                                      PASCAL
                                                                          PASCAL
          SA6     B6+2            STORE RESULT                            PASCAL
          SA1     B6+1            RETURN ADDRESS                          PASCAL
          SB7     X1                                                      PASCAL
          JP      B7                                                      PASCAL
ERR2      SX0     MESG2                                                   PASCAL
          EQ      ERR                                                     PASCAL
ERR3      SX0     MESG3                                                   PASCAL
          EQ      ERR                                                     PASCAL
ERR1      SX0     MESG1                                                   PASCAL
ERR       SA1     B6+1            RETURN ADDRESS                          PASCAL
          EQ      1778            RETURN TO MONITOR                       PASCAL
MESG1     DIS     ,Ξ* ARG TOO LARGE, ACCURACY LOSTΞ
MESG2     DIS     ,Ξ* INFINITE ARGUMENTΞ
MESG3     DIS     ,Ξ* INDEF ARGUMENTΞ
TWOPI     DATA    17175057460333447104B
DCSC1     DATA    17206220773250420550B
DCSC2     DATA    16406043230461461213B
C5        DATA    61053301014524016617B
C6        DATA    16634334143344163607B
C4        DATA    17006400637375136457B
C3        DATA    60712237223723423125B
C2        DATA    17135252525252523467B
C1        DATA    60610000000000000004B
C0        DATA    17177777777777777778B
S5        DATA    61111270663112476351B
S4        DATA    16755616534201617573B
S3        DATA    60741377140534357734B
S2        DATA    17114210421041004142B
S1        DATA    60622525252525255342B
S0        DATA    17177777777777777770B
          END
          IDENT   EXP
          ENTRY   EXP
SAVEB1    BSS     1                                                       PASCAL
EXP       SA7     B6+1            SAVE RETURN ADDRESS                     PASCAL
          SX6     B1                                                      PASCAL
          SA6     SAVEB1                                                  PASCAL
                                                                          PASCAL
          SA3     B6+3            GET ARGUMENT X                          PASCAL
                                                                          PASCAL
          MX0     0
          SA1     XMAX            GET XMAX
          SB1     1
          OR      X3,ERR2         TEST FOR BAD ARG
          ID      X3,ERR3
          SA5     A1+B1           GET XMIN
          FX7     X1-X3           XMAX-X
          SA2     A5+B1           GET LOG2(E)
          FX1     X3-X5           X-XMIN
          PX4     X0
          BX7     X7-X1           SIGN OF (XMAX-X)*(X-XMIN)
```

```
          FX6       X2*X3           X*LOG2(E)
          NG        X7,EXTR         TEST FOR ARG OUT OF (XMIN,XMAX) RANGE
          FX7       X4+X6
          DX6       X4+X6
                                                                    PASCAL
          RX7       X7+X6           .N IS NOW AN INTEGER WITH A 2000 EXPON
          NX6       B7,X7           .NORMALIZE N FOR RANGE REDUCTION  PASCAL
          SA4       A2+B1           .LOG(2) UPPER
          SA5       A4+B1           .LOG(2) LOWER
          FX0       X6*X4           .N*LOG(2) UPPER
          FX1       X6*X5           .N*LOG(2) LOWER
          FX6       X3-X0           .
          NX2       B0,X6           .
          DX3       X3-X0           .
          FX1       X3-X1           .
          FX3       X2+X1
          NX0       B0,X3           .FINAL VALUE OF X
          SB7       X7              .PICK UP N                      PASCAL
          RX7       X0*X0
          SA1       A5+B1           .C1=420.0
          SA2       A1+B1           .C0=15120.0
          FX6       X1*X7           .C1*Z
          FX5       X7*X7           .Z**2
          RX3       X6+X2           .C1*Z + C0
          SA1       A2+B1           .C3=28.0
          FX6       X1*X7           .C3*Z
          RX3       X3+X5           .C1*Z + C0 + Z*Z = B
          SA2       A1+B1           .C4=2520.0
          FX5       X0*X3           .X*B
          RX2       X6+X2           .C4 + C3*Z
          FX4       X7*X2           .Z*T
          FX1       X3+X3           .2*B
          RX6       X1-X5           .2*B - X*B
          RX1       X6+X4           .Z*T + 2*B - X*B = DENOM
          NX1       B0,X1
          RX7       X0/X1           .TERM1 = X/DENOM
          RX4       X5-X4           .X*B - Z*T = TERM2
          RX5       X7*X4           .Q = TERM1*TERM2
          SA1       A2+B1           .LOAD 1.0
          FX2       X1+X0
          DX3       X1+X0
          NX2       B0,X2
          FX4       X2+X5
          DX7       X2+X5
          RX7       X3+X7
          RX6       X4+X7
          UX7       B1,X6                                         PASCAL
          SB7       B7+B1                                         PASCAL
          PX6       B7,X6                                         PASCAL
                                                                  PASCAL
EXIT      SA1       SAVEB1                                        PASCAL
          SB1       X1                                            PASCAL
                                                                  PASCAL
          SA6       B6+2            STORE RESULT                  PASCAL
          SA1       B6+1            RETURN ADDRESS                PASCAL
          SB7       X1                                            PASCAL
          JP        B7                                            PASCAL
                                                                  PASCAL
EXTR      SX6       B0                                            PASCAL
          NG        X3,EXIT         RETURN WITH ZERO IF X < XMIN  PASCAL
          SX0       MESG1                                         PASCAL
          EQ        ER                                            PASCAL
ERR2      SX0       MESG2                                         PASCAL
          EQ        ER                                            PASCAL
ERR3      SX0       MESG3                                         PASCAL
ER        SA1       B6+1            RETURN ADDRESS                PASCAL
```

```
          EQ        177B              RETURN TO MONITOR                    PASCAL
                                                                           PASCAL
MESG1     DIS       ,Ξ* ARG TOO LARGE, FLOATING OVERFLOWΞ
MESG2     DIS       ,Ξ* INFINITE ARGUMENTΞ
MESG3     DIS       ,Ξ* INDEF ARGUMENTΞ
XMAX      DATA      741.67
XMIN      DATA      -675.82
LOG2E     DATA      1720561250731225603OB
LOG2U     DATA      1717542710277575000OB
LOG2L     DATA      1653071736325711707 3B
C1        DATA      420.0
C0        DATA      15120.0
C3        DATA      28.0
C4        DATA      2520.0
ONE       DATA      1.0
          END
          IDENT     LN                                                   PASCAL
          ENTRY     LN                                                   PASCAL
B1B2      BSS       1                                                    PASCAL
LN        SA7       B6+1              SAVE RETURN ADDRESS                PASCAL
          MX1       42                                                   PASCAL
          SX6       B1                                                   PASCAL
          BX6       -X1*X6                                               PASCAL
          LX6       18                                                   PASCAL
          SX0       B2                                                   PASCAL
          BX0       -X1*X0                                               PASCAL
          BX6       X6+X0                                                PASCAL
          SA6       B1B2                                                 PASCAL
                                                                         PASCAL
          SA1       B6+3              ARGUMENT
                                                                         PASCAL
          SA2       SQ2               . 1.414...*2.47
          UX7       B2,X1             .
          ZR        X1,ERR1
          OR        X1,ERR3
          ID        X1,ERR4
          NG        X1,ERR2
          SB7       -47               .TRY K=- 47                        PASCAL
          SB1       1
          IX6       X7-X2             .
          NG        X6,GL             .
          SB7       B7-B1             .NEED K = -48                      PASCAL
GL        PX7       B7,X7             .FORM W = 2.K*C                    PASCAL
          SA5       A2+B1             .LOAD 1.0
          FX0       X7-X5             .(W-1.0)
          NX2       B0,X0
          DX0       X7-X5
          RX0       X2+X0
          RX2       X7+X5
          RX0       X0/X2
          FX7       X0*X0             .Z=T*T
          SA1       A5+B1             .D0
          SA2       A1+B1             .D1
          SA3       A2+B1             .D2
          FX6       X7*X2             .Z*D1
          FX5       X7*X7             .Z*Z
          FX1       X1+X6             .D0 + Z*D1
          FX6       X5*X3             .D2*Z*Z
          FX4       X5*X7             .Z**3
          SA2       A3+B1             .D3
          FX1       X1+X6             .D0 + D1*Z + D2*Z*Z
          FX3       X4*X2             .D3*Z**3
          FX1       X1+X3             .TOTAL DENOMINATOR
          NX4       B0,X1
          FX6       X0/X4
          SA2       A2+B1             .C1
```

```
          SA1     A2+B1           .C2
          FX7     X7*X2           .C1*Z
          FX5     X1*X5           .C2*Z**2
          FX3     X3+X3           .2*C3*Z**3
          FX0     X0+X0           .2*T
          FX4     X7+X3           .ADD C1*Z
          SX3     B2-B7                                           PASCAL
          FX4     X4+X5           .ADD C2*Z**2
          PX1     X3
          FX7     X6*X4           .FINAL TERM OF Q
          NX5     B0,X1
          SA3     A1+B1           .LOG(2.0)
          SA2     A3+B1
          FX6     X5*X3
          FX4     X0-X7
          FX1     X5*X2
          DX5     X0-X7
          NX4     B0,X4
          RX5     X4+X5
          RX5     X5+X1
          RX5     X5+X6
          NX6     B0,X5
                                                                  PASCAL
EXIT      SA1     B1B2                                            PASCAL
          SB2     X1                                              PASCAL
          AX1     18                                              PASCAL
          SB1     X1                                              PASCAL
          SA6     B6+2            STORE RESULT                    PASCAL
          SA1     B6+1            RETURN ADDRESS                  PASCAL
          SB7     X1                                              PASCAL
          JP      B7                                              PASCAL
                                                                  PASCAL
ERR1      SX0     MESG1                                           PASCAL
          EQ      ER                                              PASCAL
ERR2      SX0     MESG2                                           PASCAL
          EQ      ER                                              PASCAL
ERR3      SX0     MESG3                                           PASCAL
          EQ      ER                                              PASCAL
ERR4      SX0     MESG4                                           PASCAL
ER        SA1     B6+1            RETURN ADDRESS                  PASCAL
          EQ      177B            JUMP TO MONITOR                 PASCAL
                                                                  PASCAL
MESG1     DIS     ,=* ZERO ARGUMENT=
MESG2     DIS     ,=* NEGATIVE ARGUMENT=
MESG3     DIS     ,=* INFINITE ARGUMENT=
MESG4     DIS     ,=* INDEF ARGUMENT=
SQ2       DATA    5520236314774736B
ONE       DATA    1.0
D0        DATA    10395.0
D1        DATA    6042103045655630403 3B
D2        DATA    1734452532634700420 1B
D3        DATA    -230.419130393980937
C11       DATA    6043116677777776772 8B
C2        DATA    1734515270155526762 7B
LOGE2     DATA    1717542710277575000 0B
          DATA    1653071736325711000 0B
          END
          IDENT   SQRT
          ENTRY   SQRT
B1B2      BSS     1                                               PASCAL
SQRT      SA7     B6+1            SAVE RETURN ADDRESS             PASCAL
          MX1     42                                              PASCAL
          SX6     B1                                              PASCAL
          BX6     -X1*X6                                          PASCAL
          LX6     18                                              PASCAL
          SX0     B2                                              PASCAL
```

```
          BX0     -X1*X0                                          PASCAL
          BX6     X6+X0                                           PASCAL
          SA6     B1B2                                            PASCAL
                                                                  PASCAL
          SA1     B6+3            GET ARGUMENT                    PASCAL
                                                                  PASCAL
          ZR      X1,OK
          NG      X1,ERR1
          SB1     1
          SB2     -48
          SA5     CA              .LOAD CA
          OR      X1,ERR2
          ID      X1,ERR3
          PX7     B2,X1           .W*2(-48)
          SA4     A5+B1           .LOAD CB
          FX0     X5*X7           .CA*W
          UX5     B7,X1           .PICK OFF N-48 AND COEFFICIENT OF W   PASCAL
          SX3     B7-B2           .N-48 -(-48) = N = 2*K + R            PASCAL
          FX2     X4+X0           .CA*W + CB = INITIAL GUESS
          AX6     X3,B1           .K = N/2
          IX5     X3-X6           .N-K
          FX4     X2*X2           .B*B
          IX3     X5-X6           .N-K-K = R
          SA5     X3+TWOR         LOAD 2.**(R/2)*2(-1)
          SB2     X6              .B2 = K
          FX0     X4+X7           .B*B + W
          FX3     X2*X0           .B*(B*B + W) = DENOM
          UX6     B7,X5           .                                     PASCAL
          SB2     B2+B7           .INCORPORATE K INTO EXPONENT OF 2**(R/2)*2
          PX6     B2,X6           . 2(K)*2(R/2)*2(-1)
          NX3     B0,X3
          FX2     X6/X3           .(2(K)*2(R/2)*2(-1) )/DENOM = TERM1
          FX5     X0*X0           .(B*B + W)**2
          FX7     X7+X7           .2*W
          FX7     X7+X7           .4*W
          FX3     X7*X4           .4*W*B*B
          FX7     X3+X5           .NUM = (B*B+W)**2 + 4*W*B*B
          FX0     X7*X2           .TERM1*NUM = 2*U
          SA5     A4+B1           .LOAD .25
          NX0     B0,X0
          FX2     X1/X0           .ARG/(2*U)
          FX3     X5*X0           . .25*(2*U)
          FX6     X2+X3           .SUM OF ABOVE = SQRT(ARG)
                                                                  PASCAL
RETURN    SA1     B1B2                                            PASCAL
          SB2     X1                                              PASCAL
          AX1     18                                              PASCAL
          SB1     X1                                              PASCAL
          SA6     B6+2            STORE RESULT                    PASCAL
          SA1     B6+1            RETURN ADDRESS                  PASCAL
          SB7     X1                                              PASCAL
          JP      B7                                              PASCAL
                                                                  PASCAL
OK        SX6     B0                                              PASCAL
          EQ      RETURN                                          PASCAL
ERR1      SX0     MESG1                                           PASCAL
          EQ      ERR                                             PASCAL
ERR2      SX0     MESG2                                           PASCAL
          EQ      ERR                                             PASCAL
ERR3      SX0     MESG3                                           PASCAL
ERR       SA1     B6+1            RETURN ADDRESS                  PASCAL
          EQ      1778            JUMP TO MONITOR                 PASCAL
                                                                  PASCAL
MESG1     DIS     ,Ξ* NEGATIVE ARGUMENTΞ
MESG2     DIS     ,Ξ* INFINITE ARGUMENTΞ
MESG3     DIS     ,Ξ* INDEF ARGUMENTΞ
```

```
CA        DATA    .585786437
CB        DATA    .4204951288
QUART     DATA    1716400000000000000018
          DATA    .353553390593
TWOR      DATA    .5
          DATA    .707106781186
          END
          IDENT   ATAN
          ENTRY   ATAN
B1B4      BSS     1                                                PASCAL
ATAN      SA7     B6+1            SAVE RETURN ADDRESS              PASCAL
          MX1     42                                               PASCAL
          SX6     B1                                               PASCAL
          BX6     -X1*X6                                           PASCAL
          LX6     18                                               PASCAL
          SX0     B4                                               PASCAL
          BX0     -X1*X0                                           PASCAL
          BX6     X6+X0                                            PASCAL
          SA6     B1B4                                             PASCAL
                                                                   PASCAL
          SA2     B6+3            GET ARGUMENT                     PASCAL
                                                                   PASCAL
          SA1     ONE
          BX0     X2
          OR      X2,ERR1         TEST FOR BAD ARG
          SA4     TANP1           LOAD TAN(PI/16)
          ID      X2,ERR2
          SB1     1
          AX2     59
          SB7     X2+777777B      SAVE SIGN IN B7
          BX3     X2-X0           X3  =  ABS(ARG)
AT1       FX7     X3-X1           .X-1
          SA5     A4+B1           .SQ2-1= .414.....
          PL      X7,LARG         .GO TO LARG IF X .GE. 1.0
          IX7     X3-X4           .X-TAN(PI/16)
          FX6     X3-X5           .X-(SQ2-1)
          SB4     B0              .FLAG = 0
          SA5     A5+B1           .LOAD TAN(3PI/16)
          NG      X7,NRR          .IF X .LT. TAN(PI/16)
          PL      X6,T2
T1        FX7     X3*X4           .X*TAN(PI/16)
          RX6     X3-X4           .X-TAN(PI/16)
          SB4     B1              .FLAG = 1
          NX0     B0,X6
          RX7     X1+X7           . 1.0 + X*TAN(PI/16)
          RX3     X0/X7           .DIVIDE TO GET R
          JP      NRR
T2        FX7     X3*X5           .X*TAN(3PI/16)
          RX6     X3-X5           .X-TAN(3PI/16)
          SB4     B1+B1           .FLAG=2
          NX0     B0,X6
          RX7     X1+X7           .1.0 + X*TAN(3PI/16)
          RX3     X0/X7           .DIVIDE TO GET R
          JP      NRR
T4        FX7     X3*X4           .X*TAN(PI/16)
          FX0     X3+X4           .X+TAN(PI/16)
          SB4     B4+B4           .FLAG = 4
          FX6     X7-X1           .X*TAN(PI/16) - 1.0
          NX5     B0,X6           .
          FX3     X5/X0           .R
          JP      NRR             .
LARG      SA2     A1-B1           .SQ2+1= 2.1414.....
          SA5     A5+B1           .TAN(3PI/16)
          FX0     X3-X2           .X-(SQ2+1)
          SB4     B1+B1           .
T3        PL      X0,T4           .
```

```
         FX7     X3*X5           .X*TAN(3PI/16)
         RX0     X3+X5           .
         RX6     X7-X1           .X*TAN(3PI/16) - 1.0
         NX5     B0,X6           .
         RX3     X5/X0           .R
         SB4     B4+B1           .FLAG = 3
NRR      RX0     X3*X3           .Z = R*R
         SA1     A5+B1           .DB3
         SA2     A1+B1           .DB2
         FX7     X0*X0           .Z**2
         FX6     X1*X0           .DB3*Z
         FX5     X6+X2           .(DB3*Z+DB2)
         SA4     A2+B1           .DB1
         FX1     X4*X0           .DB1*Z
         FX2     X5*X7           .  (  )*Z**2
         SA5     A4+B1           .DB0
         FX6     X2+X1
         FX1     X6+X5           .TOTAL DENOMINATOR
         RX6     X3/X1           .R/DENOM
         SA2     A5+B1           .DA3
         FX5     X2*X0           .DA3*Z
         SA1     A2+B1           .DA2
         SA4     A1+B1           .DA1
         FX2     X5+X1           .(DA3*Z+DA2) = ( 2)
         FX7     X7*X2           .( 2)*Z**2
         FX0     X4*X0           . DA1*Z
         SA1     A4+B1           .DA0
         FX2     X7+X0           .( 2)*Z**2 + DA1*Z
         FX2     X1+X2           .DA0 + ABOVE
         RX7     X6*X2           .REST OF SERIES
         RX2     X3-X7           .ATAN(R)
         SA4     COR0+B4         .LOAD CORRECTION TERM 1
         SA1     A4+5            .LOAD CORRECTION TERM 2
         NX2     B0,X2           .
         RX5     X2+X4           .
         NX4     B0,X5           .
         RX5     X4+X1           .
         NX3     B0,X5           .
         SX2     B7-B0
         BX6     X3-X2           .SIGN(ARG)*ATAN(ABS(ARG)  )
                                                              PASCAL
         SA1     B1B4                                         PASCAL
         SB4     X1                                           PASCAL
         AX1     18                                           PASCAL
         SB1     X1                                           PASCAL
         SA6     B6+2            STORE RESULT                 PASCAL
         SA1     B6+1            RETURN ADDRESS               PASCAL
         SB7     X1                                           PASCAL
         JP      B7                                           PASCAL
                                                              PASCAL
ERR1     SX0     MESG1                                        PASCAL
         EQ      ERR                                          PASCAL
ERR2     SX0     MESG2                                        PASCAL
ERR      SA1     B6+1            RETURN ADDRESS               PASCAL
         EQ      1778            JUMP TO MONITOR              PASCAL
                                                              PASCAL
MESG1    DIS     ,Ξ* INFINITE ARGUMENTΞ
MESG2    DIS     ,Ξ* INDEF ARGUMENTΞ
SQ2PI    DATA    2.414213
ONE      DATA    1.0
TANP1    DATA    17156272657005246138
SQ2M1    DATA    .414213
TAN3P    DATA    17175260670125337715B
DB3      DATA    17355170257740451612B
DB2      DATA    17405760164664461126B
DB1      DATA    17416476573535613424B
```

```
DB0       DATA      1741407737000000000B
DA3       DATA      1735411564321765010 3B
DA2       DATA      1737541763217502311 5B
DA1       DATA      1737537723777777770 8B
DA0       DATA      1650754451450410131 4B
COR0      DATA      0.0
          DATA      1636600000000000000B
          DATA      1715622077325042055 0B
          DATA      1716766517212252470 2B
          DATA      1717537667372335647 3B
          DATA      0.0
          DATA      1715622077325042055 0B
          DATA      1716622077325042055 1B
          DATA      1716766517212252470 3B
          DATA      1717537667372335647 4B
          END
          IDENT     FQC,C.PPFWA
          PERIPH
          TITLE     F Q C   -   FREQUENCY COUNTER
          SST
          PXAB      OPSYN     NIL
          DUMPX     OPSYN     NIL

          SYMBOL DEFINITION

CHLOCK    EQU       0
.FQCLCK   EQU       4                   LOCK BYTE



T.FQCLCK  EQU       20438               LOCK WORD (CHANNEL CHLOCK)
DELAY     EQU       20000               5 SEC DELAY IF LOCKED OUT
MAX       EQU       1600                MAX NB OF COUNTERS
C.LOWER   EQU       1                   LOWER BOUNDARY
C.UPPER   EQU       3                   UPPER BOUNDARY

          MACROS

ADX       MACRO     X                   ADD CONSTANT
          LOCAL     C
C         SET       X
IFA1      IFNE      0,C
IFA2      IFGT      C,0
IFA3      IFLE      C,778
          ADN       C
IFA3      ELSE
          ADC       C
FA3       ENDIF

IFA2      ELSE
IFA4      IFGE      C,-778
          SBN       -C
IFA4      ELSE
          ADC       C
IFA4      ENDIF
IFA2      ENDIF
IFA1      ENDIF
          ENDM

LDX       MACRO     X                   LOAD CONSTANT
          LOCAL     C
C         SET       X
IFL       IFGE      C,0
IFL2      IFLE      C,778
```

```
          LDN       C
IFL2      ELSE
          LDC       C
IFL2      ENDIF
IFL       ELSE
IFL3      IFGE      C,-77B
          LCN       -C
IFL3      ELSE
          LDC       C
IFL3      ENDIF
IFL       ENDIF
          ENDM

LDCM      MACRO     OFFSET,TABLE        LOAD CM ADDRESS
          LDD       TABLE
L1        IFEQ      TABLE,D.RA
          SHN       6
L1        ELSE
L2        IFNE      TABLE,D.CPAD
          LPN       378
          SHN       12
          ADD       TABLE+1
L2        ENDIF
L1        ENDIF
          ADX       OFFSET
          ENDM

STCD      MACRO     LOC                 STORE CM ADDRESS
          STD       LOC+1
          SHN       -12
          STD       LOC
          ENDM

          USAGE OF DIRECT CELLS

          ORG       1
LENGTH    BSS       1
I         BSS       1
PARAMS    BSS       2                   ABS ADDRESS OF PARAMETERS
ADDR      BSS       2                   ABS ADDRESS OF DUMP AREA
          ORG       D.TC+5              RES FOR MTR REQUESTS
CMWRD     BSS       5                   TEMP
BOUNDARY  EQU       CMWRD               CONTAINS BOUNDARY (2. PARAM WRD)
SIZE      EQU       BOUNDARY            CELL CONTAINING EXPONENT OF
                                        SEGMENT SIZE
LCKWRD    BSS       5                   HOLDS LOCK WORD
STAT      BSS       5                   CONTAINS STATUS WRD (FIRST PARAM)
PPIR      BSS       5                   PP INPUT REGISTER
CPSTAT    BSS       5                   CONTROL POINT STATUS

          INITIALIZE PP

          ORG       C.PPFWA
FQC       PPENTRY   PPIR,CPSTAT

          MAKE SURE THAT FQC IS ONLY USED ONCE AT ANY TIME

          LDN       CHLOCK              REQUEST CHANNEL
          RJM       R.RCH
          LDX       T.FQCLCK            GET LOCK WORD
          CRD       LCKWRD
          LDD       LCKWRD+C.FQCLCK
          PXAB      LK
          ZJN       FQC1                NO OTHER FQC

          LDN       CHLOCK              THERE IS ANOTHER FQC, DROP CHANNEL
```

```
          RJM       R.DCH
          LDD       D.PPMES1
          CWD       PPIR
          LDN       P.ZERO
          CRD       D.T0
          LDX       DELAY               REQUEST FQC ANOTHER TIME
          STCD      D.T1
          LDX       M.RPJ
          RJM       R.MTR
          LDC       WMSG                ISSUE DAYFILE MESSAGE
          RJM       R.DFM

DROP      LDX       M.DPP               DROP PP
          RJM       R.MTR
          LJM       R.IDLE

FQC1      LDN       1                   LOCK
          STD       LCKWRD+C.FQCLCK
          LDX       T.FQCLCK
          CWD       LCKWRD
          LDN       CHLOCK              DROP CHANNEL
          RJM       R.DCH

          GET PARAMETER ADDRESS AND CHECK IT

          LDD       PPIR+3
          LPN       37B
          SHN       6
          SBD       D.FL
          SHN       6
          ADD       PPIR+4
          PXAB      PA
          ADN       2
          MJN       *+3         IN USER AREA
          RJM       ERR
          SBN       2
          SHN       12
          ADD       D.FL
          ADD       D.RA                ADD REFERENCE ADDRESS
          SHN       6
          STCD      PARAMS


          GET BOUNDARIES

          LDCM      0,PARAMS
          CRD       STAT
          ADN       1
          CRD       BOUNDARY
          LDD       STAT+4              CHECK IF LEGAL CALL
          PXAB      ST
          LPN       3
          ZJN       *+3                 OK
          RJM       ERR

          LDCM      0,BOUNDARY+C.UPPER  GET UPPER BOUNDARY
          LMC       777777B             COMPLEMENT.
          PXAB      UP
          STM       BDTEST+1            INSERT IN TEST INSTRUCTION
          SHN       -12
          RAM       BDTEST

          LDD       SIZE                GET EXP OF SEGM SIZE
          LPN       7                  USE 7  BITS ONLY
          STD       SIZE                RESTORE
          LMC       1077B               MAKE NEGATIVE SHIFT INSTR
```

```
          STM       LC                  INSERT AT LC
          STM       LL1
          LMN       77B
          STM       LL2                 POSITIVE SHIFT

          LDD       BOUNDARY+C.UPPER    COMPUTE LENGTH OF AREA
          SBD       BOUNDARY+C.LOWER
          SHN       12
          ADD       BOUNDARY+C.UPPER+1
          SBD       BOUNDARY+C.LOWER+1
          PXAB      LE
          PJN       *+3
          RJM       ERR                 NEGATIVE
LL1       SHN       **        DIVIDE BY 2↑SIZE
          PXAB      L1
          STD       LENGTH
          ADX       -MAX-1              COMPARE WITH MAXIMUM
          MJN       *+3
          RJM       ERR                 TOO BIG
          LDD       LENGTH              INS LENGTH*2↑SIZE IN TEST INSTR
LL2       SHN       **
          PXAB      L2
          STM       ADDLE+1
          SHN       -12
          RAM       ADDLE

          CLEAR COUNTERS

          LDD       LENGTH
          STD       I
CL        LDN       0
          STM       CNTLOW,I
          STM       CNTHIGH,I
          SOD       I
          PJN       CL

          SET COMPLETION BIT

          AOD       STAT+4
          LDCM      0,PARAMS
          PXAB      SS
          CWD       STAT

          RECALL CENTRAL PROGRAM

          LDX       M.RCLCP
          RJM       R.MTR
          UJN       LP                  START MONITORING

          MONITOR LOOP        (34 MICROSEC)

LA.A      AOD       CMWRD               3 MICROSEC DELAY
          RPN       0
          PSN
          UJN       LA

LP        LDN       T.CPT1              CHECK IF CONTROL POINT ACTIVE
          CRD       CMWRD
          LDD       CMWRD+4
          LMD       D.CPAD
          ZJN       LA.A                CPU A IS AT THIS CTLPT
          LDD       CMWRD+3
          LMD       D.CPAD
          NJN       NA                  NO CPU - CTLPT NOT ACTIVE
          RPN       1                   ITES CPU B
```

```
A       ADC     0               CHECK IF P IS IN RANGE

BOTEST  EQU     LA              UPPER INSERTED BEFORE
        PJN     LD              P≥UPPER
ADDLE   ADC     0               ADD LENGTH, INSERTED BEFORE
        MJN     LB              P<LOWER
        UJN     LC              IN RANGE
LB      LCN     1               OUT OF RANGE
        UJN     LC+1

        INCREASE APPROPRIATE COUNTER

LC      SHN     **              DIVIDE BY 2↑SIZE
        STD     I
        AOM     CNTLOW+1,I
        SHN     -12
        ZJN     LP
        AOM     CNTHIGH+1,I       OVERFLOW
        UJN     LP

LD      PSN                     DELAY
        PSN
        UJN     LB

        CONTROL POINT NOT ACTIVE

NA      LDCM    W.CPSTAT,D.CPAD GET STATUS
        CRD     CPSTAT

        LDD     CPSTAT+C.CPEF   CHECK ERROR FLAG
        PXAB    EF
        ZJN     *+3
        LJM     DROP1           DROP IF NONZERO

        LDD     CPSTAT+C.CPSM   CHECK MOVE REQUEST
        PXAB    MF
        NJN     NA.             MOVE
        LDCM    0,PARAMS
        PXAB    PA
        CRD     STAT            GET STATUS WORD
        UJN     NA1
NA.     LDN     M.PAUSE   MACH MAL PAUSE
        RJM     R.MTR

NA0     LDCM    W.CPSTAT,D.CPAD WAIT UNTIL MOVE FLAG CLEAR
        CRD     CPSTAT
        LDD     CPSTAT+C.CPEF   OR ERROR FLAG SET
        PXAB    EF
        ZJN     *+3
        LJM     DROP1           ERROR FLAG SET
        LDD     CPSTAT+C.CPSM
        NJN     NA0             MOVE FLAG STILL SET

        LDD     CPSTAT+C.CPRA   RESTORE REFERENCE ADDRESS
        STD     D.RA

        LDD     PPIR+3          RESTORE PARAMETER ADDRESS
        LPN     378
        SHN     6
        ADD     D.RA
        SHN     6
        ADD     PPIR+4
        CRD     STAT
        PXAB    PA
        STCD    PARAMS
```

```
NA1     LDD     STAT+4          CHECK FOR DUMP REQUEST
        PXAB    DQ
        LPN     2
        NJN     *+3
        LJM     LP              GO ON MONITORING

        DUMP COUNTERS

        LDCM    2,PARAMS        GET ADDRESS OF DUMP AREA
        CRD     CMWRD
        LDCM    0,CMWRD+3
        PXAB    AA
        SHN     12
        SBD     CPSTAT+C.CPFL
        SHN     6
        ADD     LENGTH
        MJN     *+3
        RJM     ERR             AREA OUTSIDE FIELD LENGTH
        SBD     LENGTH
        SHN     12
        ADD     CPSTAT+C.CPFL
        ADD     D.RA
        SHN     6
        PXAB    AD
        STCD    ADDR

        LDX     P.ZERO          DUMP NOW
        CRD     CMWRD
        LDN     0
        STD     I
XFER    LDM     CNTHIGH,I
        PXAB    HI
        STD     CMWRD+3
        LDM     CNTLOW,I
        PXAB    LO
        STD     CMWRD+4
        LDCM    0,ADDR
        ADD     I
        CWD     CMWRD
        AOD     I
        PXAB    IX
        SOD     LENGTH
        PJN     XFER            NEXT COUNTER

        SET COMPLETION BIT

        LDD     STAT+4
        SCN     1
        ADN     1
        STD     STAT+4
        LDCM    0,PARAMS
        CWD     STAT
        LJM     DROP1           GO DROP PP

        ERROR ROUTINE

RR      ENM     X

        LDM     ERR
        LPN     7
        STM     MSG2
        LDM     ERR
        LPN     7B
        SHN     3
        ADC     2R00
        RAM     MSG2
```

```
          LDM       ERR
          SHN       12
          STM       ERR
          LPN       7
          STM       MSG1
          LDM       ERR
          LPN       70B
          SHN       3
          ADC       2RJ0
          RAM       MSG1
          LDC       MSG                 DAYFILE MESSAGE
          RJM       R.DFM
          LDX       P.ZERO              REQUEST PP ABORT
          CRD       D.T0
          LDD       D.CPAD
          SHN       -7
          STD       D.T1
          LDX       F.ERPP
          STD       D.T2
          LDX       M.SEF
          RJM       R.MTR

          DROP PP JOB

DROP1     LDX       CHLOCK              REQUEST CHANNEL
          RJM       R.RCH
          LDX       T.FQCLCK
          CRD       LCKWRD
          LDN       0
          STD       LCKWRD+C.FQCLCK     CLEAR LOCK
          LDX       T.FQCLCK
          CWD       LCKWRD
          LDN       CHLOCK              DROP CHANNEL
          RJM       R.DCH
          LJM       DROP

          DUMPX     DROP1

          MESSAGES

MSG       DATA      H*FQC ERROR AT *
MSG1      DATA      0
MSG2      DATA      0
          DATA      0
MSG       DATA      H*WAITING FOR FQC*
          DATA      0

          COUNTER AREA

CNTLOW    BSS       0
NTHIGH    EQU       CNTLOW+MAX+1



TEST      EQU       CNTHIGH+MAX+1
          END       FQC
```

FQCOUT. E. MARMIER, OCTOBER 1971.

Y POSYS, POSYS AND FQC INITIALIZE CM LOCATIONS 50018 THROUGH 10105B (CF.

AIN PROGRAM DECLARATIONS). THE PERIPHERAL PROCESSOR ROUTINE FQC SUPER-
ISES THE USER PROGRAM USING NO MORE THAN 1688 (3160B) FREQUENCY COUNTERS

ISES THE USER PROGRAM USING NO MORE THAN 1600 (3100B) FREQUENCY COUNTERS.
URING EXECUTION OF THE USER PROGRAM THE CODE AREA MONITORED BY FQC RE-

ENGTH = BLKLGTH, WHERE BLKLGTH HAS THE FORM 2↑N. ONE FREQUENCY COUNTER IS

SSIGNED TO EVERY BLOCK. BLKLGTH IS CHOSEN AS SMALL AS POSSIBLE, BUT SO

```
FCU - FCL) IS AN INTEGER MULTIPLE OF BLKLGTH.
F FCKIND = Ξ*Ξ THE LISTING OF THE USER PROGRAM IS OUTPUT MARKED WITH AS-




$T-↓

      ADDRESS = 0..2000008;
AR    FCL, FCU : ADDRESS ↱LOWER AND UPPER BOUND OF MONITORED CODE AREA↓;
      BLKLGTH : 1..64; FCKIND : CHAR;
      OUTSIDE : INTEGER ↱NUMBER OF TIMES THE P REGISTER WAS FOUND
                        NOT TO POINT INTO THE CODE AREA↓;
      COUNTERS : ARRAY [CINDEX] OF INTEGER;
      TOTAL : INTEGER; I : CINDEX;
      CTRNUMBER1 : CINDEX ↱WILL BE SET TO NUMBER OF COUNTERS - 1↓;
      FQCFILE[IN] : FILE OF CHAR ↱THE LISTING, IF ANY↓;
         ↱FQCFILE IS NON-LOCAL AND MUST BE DECLARED IN THE MAIN PROGRAM.↓

ROCEDURE OUTREAL(X : REAL);
  ↱0 ≤ X ≤ 100↓
  TYPE T02 = 0..2;
  VAR Y : REAL; N : T02;
  PROCEDURE OUT(Y : REAL; M : T02);
     BEGIN IF M = N THEN WRITE(Ξ.Ξ); WRITE( CHR(TRUNC(Y) + 27) );
        IF M < 2 THEN OUT(10*(Y - TRUNC(Y)), M+1);
     END;
  BEGIN Y := X; N := 0; WHILE Y > 1 DO BEGIN Y := Y/10; N := N+1 END;
     Y := Y + 0.0005; IF Y > 1 THEN BEGIN Y := Y/10; N := N+1 END;
     OUT(10*Y,0);
  END;

ROCEDURE PLOT;
  ↱  THIS PROCEDURE IS CALLED IF FCKIND = Ξ*Ξ. IT TAKES AS INPUT FROM
  FILE FQCFILE THE LISTING OF THE USER PROGRAM AS PROCUCED BY THE COMPI-
  LER. THE LISTING CONTAINS AN ARBITRARY NUMBER OF LINES AND AN EOL (A
  ZERO WORD). EACH LINE CONSISTS OF A BLANK, A 6-DIGIT OCTAL ADDRESS, A
  BLANK, A TEXT OF LENGTH ≥ 0, AND AN EOL. IF PLOT FINDS THE SECOND CHAR-
  ACTER IN A LINE TO BE BLANK, WHICH MEANS THE USER ILLEGALY USED THE
  $C+ COMPILER INSTRUCTION, PLOT ISSUES AN ERROR MESSAGE AND COPIES THE
  REST OF THE FILE ONTO OUTPUT.

  FQCOUT GOES THROUGH FQCFILE EXACTLY ONCE.
  THE KEY VARIABLES USED ARE LADR1, LADR2, BLKADR1 AND BLKADR2. WHENEVER
  PLOT PROCESSES A LISTING LINE, LADR1 = (BINARY VALUE OF) THE ADDRESS
  ON THE LINE, LADR2 = ADDRESS ON THE NEXT LINE, BLKADR1 = ADDRESS OF THAT
  CODE BLOCK WHICH CONTAINS THE BEGINNING OF THE CODE THE LINE TRANSLATED
  INTO, AND BLKADR2 = BLKADR1 + BLKLGTH. WE THUS HAVE BLKADR1 ≤ LADR1 ≤ LADR2
  AND LADR1 < BLKADR2, EXCEPT AT THE BEGINNING WHEN LADR1 = FCL
  AND LADR2 < FCL.

  IF LADR1 ≥ LADR2 NO ASTERISK IS WRITTEN (THE LINE TANSLATED INTO NO SU-
  PERVISED CODE). OTHERWISE TWO CASES ARISE: IF LADR2 ≤ BLKADR2 THE LINEΞS
  TRANSLATION LIES ENTIRELY IN THE BLOCK DELIMITED BY BLKADR1 AND BLKADR2.
  THE FREQUENCY COUNT ASSIGNED TO THE LINE IS THAT BLOCKΞS FREQUENCY COUN-
  TER VALUE * (LADR2 - LADR1) / BLKLGTH. IF BLKADR2 < LADR2 THE CODE PRO-
  DUCED BY THE LINE OCCUPIES, SAY, W1 WORDS IN A FIRST BLOCK, 0 OR MORE
  SUBSEQUENT BLOCKS ENTIRELY, AND W2 WORDS OF A LAST BLOCK. THE FREQUEN-
  CY COUNT ASSIGNED TO THE LINE IS THEN
     FREQUENCY COUNT OF FIRST BLOCK * W1/BLKLGTH
   + SUM OF FREQUENCY COUNTS OF ALL BUT THE LAST SUBSEQUENT BLOCKS
   + FREQUENCY COUNT OF LAST BLOCK * W2/BLKLGTH.

  AN ASTERISK IS PRINTED DEPENDING ON THE RESULTING FREQUENCY COUNT OF
  THE LINE MODIFIED BY A SCALING FACTOR.                              ↓

  CONST WIDTH = 47;
  TYPE  STARPOS = 0..WIDTH;
  VAR MAX : INTEGER; SCALE : REAL; ↱MAXIMUM COUNTER VALUE; SCALING FACTOR↓
     VAL, TVAL : INTEGER; POS : STARPOS;
     LADR1, LADR2 : ADDRESS; ↱LINE ADDRESSES↓
     BLKADR1, BLKADR2 : ADDRESS; ↱BLOCK DELIMITING ADDRESSES↓
     I : CINDEX; CHCNT : 0..137; ↱INDEX FOR ARRAY OF COUNTERS; CHARACTER COUNT↓
     A : ARRAY [1..6] OF CHAR; K : 1..6; ↱FOR HOLDING ADDRESSES CHARACTERWISE↓

  PROCEDURE FINDSCALE;
     ↱  THIS PROCEDURE RETURNS A SCALING FACTOR THROUGH THE VARIABLE
     SCALE. THE SCALING FACTOR HAS THE PROPERTY THAT 1 PRINT POSITION
     CORRESPONDS TO 2↑Q OR 2↑(-Q) PERCENTS FOR SOME Q. SCALE IS MADE
     TO BE ≥ LOWSC, WHERE THE VALUE ASSIGNED TO LOWSC WAS FOUND EX-
     PERIMENTALLY.↓
     CONST FACTOR = 0.2;
```

```
    CONST FACTOR = 0.2;
    VAR SC, LOWSC : REAL;
        K : 0..137; P, PP : REAL;
    BEGIN LOWSC := WIDTH*FACTOR*(BLKLGTH + BLKLGTH DIV 3) / (MAX + 1);
        SC := 25/TOTAL;
        WHILE SC < LOWSC DO SC := 2*SC;
        SCALE := SC; P := 100/(SC*TOTAL);
        WRITE(= 1 POSITION CORRESPONDS TO =);
        OUTREAL(P); WRITE(= 0/0=); WRITE(EOL);
        P := 10*P; PP := 0;
        WRITE(=0=); FOR K := 1 TO 79 DO WRITE(= =);
        FOR K := 0 TO 4 DO
        BEGIN WRITE(=      =); OUTREAL(PP); PP := PP + P;
        END;
        WRITE(EOL);
        FOR K := 1 TO 68 DO WRITE(= =);
        WRITE(=PERCENTAGE SCALE:   =); FOR K := 1 TO 4 DO WRITE(=I----I----=);
        WRITE(=I----I--=); WRITE(EOL,=0=);
    END;

BEGIN ↑PLOT↓
    RESET(FQCFILE);
    MAX := 0;
    FOR I := 0 TO CTRNUMBER1 DO
        IF MAX < COUNTERS[I] THEN MAX := COUNTERS[I];
    FINDSCALE;

    LADR1 := FCL;
    ↑READ FIRST LISTING ADDRESS INTO LADR2: ↓
    LADR2 := 0;
    FOR K := 1 TO 6 DO
    BEGIN GET(FQCFILE); A[K] := FQCFILE↑;
        LADR2 := 8*LADR2 + ORD(FQCFILE↑) - 27;
    END;
    I := 0; BLKADR1 := FCL; BLKADR2 := FCL + BLKLGTH;

    ↑ MAIN LOOP WHICH GOES LINE BY LINE THROUGH FQCFILE: ↓
    REPEAT WRITE(EOL);
        WRITE(= =,A[1],A[2],A[3],A[4],A[5],A[6]); ↑ADDRESS OF LINE↓
        GET(FQCFILE); CHCNT := 7;
        ↑COPY REST OF LINE: ↓
        REPEAT WRITE(FQCFILE↑); GET(FQCFILE); CHCNT := CHCNT + 1;
        UNTIL FQCFILE↑ = EOL;
        ↑GET NEXT LADR2: ↓
        GET(FQCFILE);
        IF FQCFILE↑ = EOL THEN LADR2 := FCU ELSE
        BEGIN
            GET(FQCFILE);
            IF FQCFILE↑ = = = THEN ↑ILLEGAL USE OF $C+↓
            BEGIN
                WRITE(EOL,= =); FOR CHCNT := 2 TO 47 DO WRITE(=-=);
                WRITE(= $C+ NOT ALLOWED IF OPT = FC*. =);
                FOR CHCNT := 76 TO 121 DO WRITE(=-=); WRITE(EOL);
                WRITE(= =); GOTO 1;
            END;
            A[1] := FQCFILE↑; GET(FQCFILE); A[2] := FQCFILE↑;
            GET(FQCFILE); A[3] := FQCFILE↑; GET(FQCFILE); A[4] := FQCFILE↑;
            GET(FQCFILE); A[5] := FQCFILE↑; GET(FQCFILE); A[6] := FQCFILE↑;
            LADR2 := (((( ORD(A[1]))*8 + ORD(A[2]) )*8 + ORD(A[3]) )*8
                        + ORD(A[4]) )*8 + ORD(A[5]) )*8 + ORD(A[6])
                    - 36666638;
        END ↑GET NEXT LADR2↓;

        IF LADR2 > LADR1 THEN ↑LINE TRANSLATED INTO SUPERVISED CODE↓
        BEGIN
            IF LADR2 ≤ BLKADR2 THEN ↑THIS BLOCK CONTAINS WHOLE LINE TRANSLATION↓
            BEGIN
                VAL := TRUNC( COUNTERS[I]*(LADR2 - LADR1)/BLKLGTH + 0.5 );
                IF LADR2 = BLKADR2 THEN
                BEGIN BLKADR1 := BLKADR2; I := I+1;
                    BLKADR2 := BLKADR2 + BLKLGTH;
                END;
            END ELSE
            BEGIN
                ↑PARTIAL VALUE FROM FIRST BLOCK: ↓
                TVAL := COUNTERS[I] * (BLKADR2 - LADR1);
                BLKADR2 := BLKADR2 + BLKLGTH; I := I+1;
                VAL := 0;
                WHILE LADR2 ≥ BLKADR2 ↑BLOCK IS ENTIRELY COVERED
                BY TRANSLATION OF LINE↓ DO
                BEGIN VAL := VAL + COUNTERS[I]; I := I+1;
                    BLKADR2 := BLKADR2 + BLKLGTH;
                END;
                BLKADR1 := BLKADR2 - BLKLGTH;
                ↑PARTIAL VALUE FROM LAST BLOCK: ↓
                TVAL := TVAL + COUNTERS[I] * (LADR2 - BLKADR1);
                VAL := VAL + TRUNC(TVAL/BLKLGTH + 0.5);
            END;
```

```
            POS := TRUNC(VAL * SCALE + 0.5);
            IF POS ≠ 0 THEN
            BEGIN
               FOR CHCNT := CHCNT + 1 TO 88 DO WRITE(Ξ Ξ);
               IF POS > WIDTH THEN
               BEGIN FOR CHCNT := 89 TO 126 DO WRITE(Ξ*Ξ); WRITE(Ξ Ξ);
                  OUTREAL(100*VAL/TOTAL); WRITE(Ξ 0/0Ξ);
               END ELSE
               FOR CHCNT := 1 TO POS DO WRITE(Ξ*Ξ);
            END;
            LADR1 := LADR2;
         END ↱LINE TRANSLATED INTO SUPERVISED CODE↳;
      UNTIL LADR2 ≥ FCU;

      ↱COPY REST OF FQCFILE:↳
      WRITE(EOL,Ξ Ξ); IF FQCFILE↑ ≠ EOL THEN FOR K := 1 TO 6 DO WRITE(A[K]);
   1: WHILE FQCFILE↑ ≠ EOL DO
      BEGIN WRITE(FQCFILE↑);
         REPEAT GET(FQCFILE); WRITE(FQCFILE↑); UNTIL FQCFILE↑ = EOL;
         GET(FQCFILE);
      END;
      WRITE(Ξ Ξ,EOL);
   END ↱PLOT↳;

ROCEDURE OUTCOUNTERS;
   VAR A,A1,D : ADDRESS; I : CINDEX;
      K : 1..6; ALF : ARRAY [1..6] OF CHAR;
      NEWLINE : BOOLEAN;
   BEGIN WRITE(Ξ1Ξ,EOL);
      D := 4 * BLKLGTH;
      WRITE(Ξ Ξ);
      NEWLINE := TRUE; A := FCL;
      FOR I := 0 TO CTRNUMBER1 DO
      BEGIN
         IF I MOD 4 = 0 THEN
         BEGIN A1 := A; A := A+D;
            FOR K := 6 DOWNTO 1 DO
            BEGIN ALF[K] := CHR(A1 MOD 8 + 27);
               A1 := A1 DIV 8;
            END;
            IF NEWLINE THEN WRITE(EOL,Ξ Ξ) ELSE WRITE(Ξ  Ξ);
            NEWLINE := ¬NEWLINE;
            FOR K := 1 TO 6 DO WRITE(ALF[K]);
         END;
         WRITE(COUNTERS[I]);
      END;
      WRITE(EOL);
   END ↱OUTCOUNTERS↳;

EGIN ↱MAIN PROGRAM↳
```

```
   WRITE((Ξ BLKLGTH:Ξ),BLKLGTH,EOL);
```

```
  CTRNUMBER1 := (FCO - FCL) DIV BLKLGTH + 1;
  TOTAL := OUTSIDE; FOR I := 0 TO CTRNUMBER1 DO TOTAL := TOTAL + COUNTERS[I];
  WRITE((= TOTAL:=),TOTAL,EOL); WRITE((= OUTSIDE: =),OUTSIDE,= =,=(=);
  IF TOTAL # 0 THEN
  BEGIN OUTREAL(100*OUTSIDE/TOTAL); WRITE(= 0/0)=); WRITE(EOL);
  END;
  IF FCKIND = =*= THEN PLOT ELSE OUTCOUNTERS;
ND.
        IDENT   PMD
        ENTRY   PMD

 PMD IS LOADED BY THE MONITOR AS AN OVERLAY. PREVIOUS TO ITS CALL, THE
 MONITOR LOADS AN ABBREVIATED CONTEXT-TABLE INTO THE INPUT BUFFER TO BE USED
 BY PMD. THE REGISTERS ARE LEFT IN THE FOLLOWING CONFIGURATION:
        B1 = 1
        B5 = Y,THE POINTER TO THE CURRENTLY ACTIVE BLOCK OF THE RT-STACK
        B6 = X,ADDRESS FOR WHICH PMD SEEKS A CORRESPONDENCE IN THE TABLE
        X5 = BT,ABS. ADR. OF WORD IMMEDIATELY PRECEDING FIRST ENTRY IN TABLE
        X6 = ET,ABS. ADDRESS OF THE LAST WORD OF THE  TABLE
        X7 = RETURN ADDRESS

WRITEOCT EQU    111B
TEXT    EQU     112B            *CALLED AT ONLY ONE POINT
INTGR   EQU     114B
EREAL   EQU     115B
ALF     EQU     117B
CHAR    EQU     120B
BOOL    EQU     121B

CALL    MACRO   P+1
        EQ      P
        ENDM

GFIELD  MACRO   XI,BJ,N         *GETS A FIELD(DEPENDENT ON N)OF X1
        S*BJ    N
        AX0     BJ,XI           *AND STORES IT IN BJ
        S*BJ    X0
        ENDM

NLINE   MACRO                   *WRITES AN EOL
        BX1     X0-X0
        SX2     B1
        CALL    CHAR            *WRITE CHAR
        ENDM

CASE    MACRO                   *MOVES THE POINTER TO THE NEXT RECORD
        SB4     B4+2
        SA4     B4
        ENDM

RNEQV   BSS     1
NEQV    SA7     RNEQV           *WRITES <NAME> = <VALUE>
        NLINE
        SA1     B4-1            *GET NAME
        SX2     17
        CALL    ALF
        SX1     54B             *=:=
        SX2     1
        CALL    CHAR
        SX1     1R
        SX2     1
        CALL    CHAR
        SA4     B4
        SB2     X4              *GET RELATIVE ADDRESS OF VALUE
        SA1     B5+B2           *COMPUTE ABSOLUTE ADR AND STORE VALUE IN X1
        GFIELD  X4,B3,40        *IS THIS A VAR PARM.
        EQ      B3,DIRECT
        SA1     X1
        SB3     A1-B5
        GE      B3,TUMMY
        SB3     5000B
        SB3     A1-B3
        LT      B3,TUMMY
DIRECT  GFIELD  X4,B3,22        *GET THE TYPE
        BX7     RNEQV
        SA2     SWITCH+B3-1
        SB3     X2
        AX2     18
        JP      B3              *WRITE VALUE
ALFS    LX1      6
        ZR       X1,RETALF
        MX0      54
        BX6      X0*X1
        SA6      SVALF
        BX1      -X0*X1
        SX2      B1
        NZ       X1,CHAR
        SX1      1R                     *WRITE A BLANK FOR EOL=S
        EQ       CHAR
```

```
          EQ        CHAR
          SA1       SVALF
          EQ        ALFS
RETALF    SA1       RNEQV
          SB7       X1
          JP        B7
CHARR     ZR        X1,CEOL
          BXJ       X1
          AX0       6
          ZR        X0,CHAR
          SA1       UNDEF
          SX2       7
          EQ        ALF
CEOL      SX2       3
          SA1       EOL             ↑ ΞEOLΞ
          EQ        ALF
 CODE FOR POINTER TYPES. PRINTS ΞNILΞ OR THE OCTAL ADDRESS
PT        SX0       X1-200000B
          NZ        X0,WRITEOCT
          SA1       NIL
          EQ        ALF

 POST-MORTEM-DUMP IS AN  AUTOPSY OF THE USER PROGRAM FOLLOWING A FATAL ERROR
 FOR WHICH THE MONITOR REGAINS CONTROL. THE BODY LIES BELOW

 THE HEAD---STORES INFORMATION
PMD       SA7       RADR            ↑SAVE RETURN ADDRESS
          SA6       SVET            ↑SAVE ADR OF LAST ENTRY OF TABLE
          SA4       X6
          SB4       X4              ↑SET POINTER TO REL. ADR. OF (MAIN)
          SB1       1               ↑THIS SHOULD SOON BE STANDARD
          BX7       X5
          SA7       SVBT            ↑SAVE ADR OF FIRST ENTRY OF TABLE
          SX1       FIRST
          CALL      TEXT
          SX1       B6
          SX2       6
          CALL      WRITEOCT        ↑LOCATION OF THE TRAP
          NLINE
          SA5       SVBT

 LOOP BEGINS HERE. IF PMD GETS CARRIED AWAY, IT LOSES ITS HEAD.
 THE SHOULDERS BEAR THE WEIGHT
SHODR     SA4       B4+X5           ↑GET ENTRY FOR MAIN PROGRAM
          GFIELD    X4,B3,22        ↑GET ITS STARTING ADR
          LE        B3,B6,FEET      ↑WEΞVE REACHED THE BOTTOM OF THE STACK
S1        GFIELD    X4,B3,40        ↑GET LINK
          SB4       B3+X5           ↑MOVE THE POINTER TO THE NEXT PROCEDURE
          SA4       B4              ↑GET ENTRY
          GFIELD    X4,B3,22        ↑GET SA
          GT        B3,B6,S1        ↑UNTIL CORRECT INTERVAL IS LOCATED
          SX0       B1
          SB3       X4-2            ↑GET PREVIOUS COUNT - 2
          IX6       X0+X4           ↑UPDATE THE COUNT
          SA6       B4
          SA3       B5+B1
          SB6       X3-1            ↑NEW X
          GT        B3,LG1          ↑OUTPUT EACH PROCEDURE AT MOST 3X
          NLINE
          SA1       B4-1
          SX2       12
          CALL      ALF             ↑ΞPROCEDURE NAMEΞ
          SA1       ST1
          SX2       4
          CALL      ALF             ↑Ξ AT:Ξ
          SX3       B6
          LX3       45              ↑XXXXX0---0X
          MX0       57
          SA1       OCTA            ↑X1=Ξ00000000 0Ξ
          SB2       -6
S2        BX2       -X0*X3          ↑THIS LOOP STORES AN ADDRESS AS AN ALFA VAL.
          IX1       X1+X2
          LX3       3
          LX1       6
          SB2       B2+B1
          LT        B2,S2
          LX1       12
          SX2       7
          CALL      ALF             ↑Ξ<RETURN ADDRESSΞ>Ξ
          CASE
          NG        X4,LEG

 THE STOMACH REGURGITATES
          SX7       TUMMY
          EQ        NEQV            ↑AT LEAST ONE VALUE
TUMMY     SX7       *
          CASE
          PL        X4,NEQV
```

```
 THE LEGS WALK THROUGH THE STACK, REPEATING THE ABOVE, UNTIL EMPTIED
LEG     NLINE
LG1     SA3     B5+B1           ↱X3=0---0<DL><RA>
        AX3     18
        SB5     X3              ↱B5=NEW Y
        SA5     SVBT            ↱FETCH BT AND STORE IN X5
        SA1     SVET            ↱FETCH ET
        SA1     X1
        SB4     X1              ↱RESET POINTER
        NE      B5,SHODR        ↱IF B5=0 THEN B5=BASE ADR FOR GLOBALS

 HAVING COMPLETED THE LAST LEG,WE SEE THAT THE FEET(GLOBALS) ARE STILL
 KICKING AROUND
FEET    SB4     X5+B4           ↱SET POINTER TO JUST BEFORE THE GLOBALS
        NLINE
        SB5     B0
        SA1     ST2
        SX2     7
        CALL    ALF             ↱Ξ(MAIN)
+       SX7     *
        CASE
        PL      X4,NEQV

 THE FEET STEP FROM THIS FEAT
        NLINE
        SA4     RADR
        SB7     X4
        JP      B7

RADR    BSS     1
SVET    BSS     1
SVBT    BSS     1
SVALF   BSS      1
FIRST   DIS     ,Ξ1PROGRAM TERMINATED AT:  Ξ
OCTA    DATA    10L00000000 0
EOL     DATA    10LEOL
NIL     DATA    10LNIL
UNDEF   DATA     10L<UNDEF>
ST1     DATA    10L AT:
ST2     DATA    10L (MAIN)
SWITCH  VFD     42/10,18/INTGR
        VFD     42/20,18/EREAL
        VFD     42/10,18/ALFS
        VFD     42/1,13/CHARR
        VFD     42/6,18/BOOL
        VFD     42/6,18/PT

AND MAY IT REST IN PEACE
        END                     ↱AMEN
```

P A L C A L T O C Y B E R N E T C E N T E R

HOURS AVAILABLE FOR CUSTOMER SERVICE --

TERMINAL ACCESS
MN0300-2400 TU0800-2400 WD0300-2400 TH0800-2400
FR0800-2400 SA0800-1700 CLOSED SUNDAY

OVER THE COUNTER
MN0800-2400 TU0000-2400 WD0000-2400 TH0000-2400
FR0000-2400 SA0800-1700 CLOSED SUNDAY
THE ABOVE TIMES ARE SUBJECT TO CHANGE WITH 24 HOUR
NOTIFICATION.

DUE TO THE INCREASED WORKLOAD WE HAVE EXPERIENCED RECENTLY,
OUR NORMAL HOURS OF AVAILABILITY TO TERMINAL USERS MUST REVERT TO
OUR PREVIOUS SCHEDULE.THIS IS REFLECTED ABOVE.
WE ASK ANY USERS WHO HAVE SPECIAL REQUIREMENTS DURING THE
HOURS OF 0000-0800 TO MAKE ARRANGEMENTS WITH M.L.FRAZIER
DURING THE DAY PREVIOUS.WE WILL ATTEMPT TO HONOR THESE REQUESTS
WHENEVER POSSIBLE.THANK YOU.

FOR OPERATIONS AND SCHEDULING ASSISTANCE, CALL
  DAYS: (415) 493-8920 EXT 421
  NITES: (415) 493-7520, 493-7280, 493-7480

FOR PROGRAMMING ASSISTANCE, CALL
(415) 493-8920 EXT 249

PERMANENT FILE AND I/O QUEUE STATUS--
ALL PERM FILES ARE INTACT.
ALL INPUT/OUTPUT QUEUES ARE INTACT.

****** IMPORTANT NOTICE TO ALL USERS **********

SORTMRG, UPDATED TO LEVEL 340, WILL BE INSTALLED
ON JUNE 26, 1973.

FORTRAN EXTENDED (FTN), UPDATED THROUGH PSR SUMMARY 348,
WILL BE INSTALLED ON JUNE 28, 1973.

SIS, UPDATED THROUGH PSR SUMMARY 340,
WILL BE INSTALLED ON JUNE 26, 1973.

****

```
  10/03/73  PADC 6600-S/N 0065  LEVEL R   08/15/73
09.21.16.4.F20SBK2
09.21.16.IP   000001 INPUT UNITS USED.
09.21.16.$SEQUENCE,20S.
09.21.16.$CHARGE,F3037  .
09.21.16.PRINT,CM12000,T50,P4,MT1.
09.21.16.REQUEST,PASCAL,HI. (PASCAL,NO RING)
09.21.22.COPYBF(PASCAL,OUTPUT)
09.21.22.MT 37 ASSIGNED TO PASCAL
09.22.09. VSNPASCAL
09.22.10. MT 37 VISUAL REEL NUMBER IS  PASCAL
09.22.10.FILE OPENED - OUTPUT
09.22.18.OP   000283 STORAGE DATA BLOCKS ON FILE  OUTPUT
09.22.18.IP   000001 STORAGE DATA BLOCKS ON FILE  INPUT
09.22.18.CM   012000      CM CELLS USED.
09.22.18.CP   000000.024 CP SEC. USED.
09.22.18.IO   000007.336 IO SEC. USED.
09.22.18.SS   000000.330 SYSTEM SEC. USED.
09.22.18.AC -  END OF JOB.
```