

```

//      PALHD          LAST MODIFIED ON FRIDAY, 12 JUNE 1970
//      AT 5:13:24.80 BY R MABEE

// LISTING OF PAL HEADFILE AND BCPL/360 BASIC HEADFILE GOTTEN
// WITHIN SUPPRESSED BY NCLIST DIRECTIVE. TO OVERRIDE DIRECTIVE,
// SPECIFY ALLSOURCE OPTION TO BCPL COMPILER.
>>> NCLIST
>>> EJECT
      //
      //      *****
      //      *
      //      *          PALHD          *
      //      *
      //      * (COMPATIBLE WITH PALSYS) *
      //      *
      //      *****
      //

// GET BASIC BCPL/360 HEAD FILE
>>> GET 'BASIC'

MANIFEST // VECTOR APPLICATION
$( H1=0; H2=1; H3=2; H4=3; H5=4 $)

MANIFEST // SYNTACTIC OPERATORS
$( BAR=101; END=102; WHERE=103; DOT=105
  BRA=106; KET=107; IN=108; PERCENT=109
  IFSQ=110; IFNOT=111; M_DO=112      $)

MANIFEST // AE TREE NODES
$( DEF=121
  M_LET=122; LAMBCA=123; M_VALOF=124; M_TEST=125
  M_IF=126; M_WHILE=127; ASS=128
  SEQ=130; COLON=131
  NOSHARE=133; COND=134
  COMMA=137; VALDEF=138
  REC=139; M_AND=140; WITHIN=141
  MPT=142; PAREN=143      $)

MANIFEST //AE NODES AND POCODE SYMBOLS
$( M_GOTO=148; M_RES=149
  M_NOT=151; M_NIL=152; STRINGCONST=153; NAME=154
  M_PLUS=157; M_MINUS=158
  M_AUG=160; M_LOGOR=161; M_LOGAND=162
  M_GE=163; M_NE=164; M_LE=165; M_GR=166; M_LS=167; M_EQ=168
  M_MULT=169; M_DIV=170; M_POWER=171
  M_PQS=173; M_NEG=174; M_APPLY=175      $)

MANIFEST // POCODE SYMBOLS
$( M_LOADL=181; M_LOADR=182; M_LOADE=183; M_LOADS=184; M_LOADN=185
  M_RESTOREE1=187; M_LOADGUESS=188
  M_FORMCLOSURE=189; M_FORMLVALUE=190; M_FORMRVALUE=191
  M_MEMBERS=192
  M_JUMP=195; M_JUMPF=196; M_SAVE=197; M_RETURN=198
  M_TESTEMPTY=199; M_LOSE1=200; M_UPDATE=201

```

```

M_DECLNAME=203; M_DECLNAMES=204; M_INITNAME=205; M_INITNAMES=206
M_DECLLABEL=207; M_SETLABES=208; M_BLOCKLINK=209; M_RESLINK=210
M_SETUP=211
INTEGER=213; LAB=214; PARAM=215; EQU=216      $)

```

```

MANIFEST // AE NODES, POCODE SYMBOLS AND RUN-TIME NODE TYPES
$( M_DUMMY=220; JJ=221; M_TRUE=222; M_FALSE=223
  NUMBER=224; M_TUPLE=225      $)

```

```

MANIFEST // TRANSLATION SYMBOLS
$( VAL=0; REF=1 $)

```

```

MANIFEST // LENGTH OF ACTIVE INPUT FIELD IN INPUT RECORD
$( LINET = 72 $)

```

```

GLOBAL // PLACEMENT SET BY PALSYS
$( PAL:184; TIMEOVFL:199; TIME_EXCEEDED:93 $)

```

```

GLOBAL // COMPILER FUNCTIONS
$( NEXTSYMB:300; KIND:301; RCH:302; LOOKUPWORD:303; CAE:304;
  RCCMLOOP:305; REPORT:306; RCOM:307; REXP:308; RBDEF:309; RDEF:310;
  RBV:311; RDNAMELIST:312; RDNAME:313; RDNS:314; RARG:315; RDBEXP:316;
  PLIST:317; NODETYPE:318; TRANS:319; FINDLABELS:320; TRANSLABELS:321;
  TRANSRHS:322; C_DECLNAMES:323; LOADDEFINEE:224; DECLGUESSES:325;
  C_INITNAMES:326; TRANSSCOPE:327; MAPF:328; MAPB:329; C_LENGTH:330;
  NEXTPARAM:331; UPSSP:332; NEWVEC:333; LIST1:334; LIST2:335;
  LIST3:336; LIST4:337; CCMLAB:338; OUTOP:339; OUTN:340; OUTP:341;
  OUTNAME:342; OUTNUMBER:343; OUTSTRING:344; OUTPSOP:345; OUT1:346;
  CUT2:347; OVERFLOW:348 $)

```

```

GLOBAL // COMPILER GLOBAL VARIABLES
$( SYMBV:360; SYMBP:361; SYMB:362; LINEP:363; EOP:364; EOPLEVEL:365;
  CHKIND:366; NAMECHAIN:367; DUMMYN:368; PARAMNUMBER:369; SSP:370;
  MSP:371; AETREEP:372 $)

```

```

GLOBAL // VARIABLES COMMON WITH PALSYS
$( CH          : 218 // LAST CHARACTER READ
  CODEFILE     : 219 // POINTER TO POCODE STORAGE AREA
  CODEFILEP    : 220 // POINTER TO NEXT WORD POCODE STORAGE
  COMPERROR    : 229 // PRESENT JOB SEGMENT ERROR FLAG
  INPUT        : 234 // PRESENT INPUT STREAM
  LISTING      : 248 // INDICATES IF POCODE LISTING DESIRED
  LVCH         : 251 // LVALUE OF CH
  NCODE        : 257 // INDICATES IF POCODE TO BE RETAINED
  STACKWARNING : 269 // APPROXIMATE END BCPL RUN TIME STACK
  STORAGE      : 272 // POINTER TO USABLE FREE STORAGE
  STORAGET     : 273 // POINTER TO END OF USABLE FREE STORAGE
  TREE         : 286 // PAL AE TREE PRINT DEPTH
                                                    $)

```

```
>>> LIST
```

PALO

9/01/70

1:48:41

```
// PALO LAST MODIFIED ON FRIDAY, 12 JUNE 1970
// AT 5:37:14.04 BY R MABEE
>>> FILENAME 'PALO'
```

```
//
// *****
// * PALO *
// * *
// *****
//
```

```
>>> GET 'PALHD'
>>> EJECT
// PALO
```

LET PAL() BE

```
$(1 CONTROL(OUTPUT,3)
WRITES('*TPAL MK 5 ENTERED*N' )
CONTROL(OUTPUT, 2)
$( LET A=CAE()
CONTROL(OUTPUT,3)
WRITES('*TSYNTAX TREE SIZE = ')
WRITEN(STORAGET - AETREEP)
WRITECH(OUTPUT, '*N')
UNLESS TREE=0 DO $( CONTROL(OUTPUT, -1)
WRITES('SYNTAX TREE:*N*N')
PLIST(A, 0, TREE)
WRITECH(OUTPUT, '*N') $(
CONTROL(OUTPUT, 3)
IF COMPERROR LOGOR NCODE DO RETURN
PARAMNUMBER := 0
IF LISTING DO $( CONTROL(OUTPUT, -1)
WRITES('THE POCODE IS:*N') $(
$( LET N = NEXTPARAM()
EOPLEVEL := LEVEL()
SSP, MSP := 0, 1
CUTOP(M_SETUP); OUTP(N)
TRANSLABELS(A)
TRANS(A, VAL)
UNLESS SSP=1 DO WRITES('*N*N*T***** SSP ERROR*N')
OUTPSOP(N, EQU, MSP)
ECP: IF LISTING DO $( WRITECH(OUTPUT, '*N'); CONTROL(OUTPUT,3) :$)1
```

```
//      PAL1      LAST MODIFIED ON FRIDAY, 12 JUNE 1970
//      AT 5:37:14.52 BY R MABEE
>>>  FILENAME 'PAL1'
```

```
//
//      *****
//      *          *
//      *  PAL1  *
//      *          *
//      *****
//
```

```
>>>  GET 'PALHD'
```

```
>>>  EJECT
// PAL1A
```

```
MANIFEST $( EMPTY=0; SIMPLE=1; IGNOREABLE=2; OTHERS=3
           DOTK = 4; CAPITAL = 6; DIGIT = 7  $)
```

```
LET NEXTSYMB() BE
$(1 LET DIG = FALSE
    SYMBP := 0
    UNLESS CHKIND = EMPTY GOTO M
L:  RCH()
M:  CHKIND := KIND()
    SWITCHCN CHKIND INTO

        $( CASE IGNOREABLE:  RCH() REPEATWHILE CH='*S'
            GOTO M

DIGITRDR:
CASE DIGIT: SYMBP := SYMBP+1
            DIG := TRUE
            SYMBV*(SYMBP) := CH
            RCH()
            CHKIND := KIND()
            SWITCHCN CHKIND INTO

                $( CASE DIGIT:      GOTO DIGITRDR
                    CASE CAPITAL:   GOTO IDRDR
                    CASE DOTK:      GOTO NUMBEG
                    DEFAULT:        SYMB := NUMBER
                                    RETURN $)

IDRDR:
CASE CAPITAL:
    SYMBP := SYMBP+1
    SYMBV*(SYMBP) := CH
    RCH()
    CHKIND := KIND()
    IF CHKIND GE 5 DO
```

```

$( IF CHKIND = DIGIT DO DIG := TRUE
   GOTO IORDR $)
SYMB := DIG -* NAME, LOOKUPWORD()
RETURN

```

```

NUMBEG: SYMBP := SYMBP + 1
        SYMBV*(SYMBP) := CH
        RCH()
        CHKIND := KIND()
        IF CHKIND = DIGIT GOTO NUMBERDR
        REPORT(5, 1, 'INCORRECT REAL')
        SYMB := NUMBER
        RETURN

```

```

NUMBERDR:
        SYMBP := SYMBP + 1
        SYMBV*(SYMBP) := CH
        RCH()
        CHKIND := KIND()
        IF CHKIND = DIGIT GOTO NUMBERDR
        SYMB := NUMBER
        RETURN

```

```

CASE DOTK:   CHKIND, SYMB := EMPTY, DCT; RETURN

```

```

CASE SIMPLE: CHKIND := EMPTY
             $)

```

#### SWITCHON CH INTO

```

$( CASE ';' : SYMB := SEQ; RETURN
   CASE ',' : SYMB := CCMMA; RETURN
   CASE '+' : SYMB := M_PLUS; RETURN
   CASE '(' : SYMB := BRA; RETURN
   CASE ')' : SYMB := KET; RETURN
   CASE '=' : SYMB := VALDEF; RETURN
   CASE '&' : SYMB := M_LOGAND; RETURN
   CASE '$' : SYMB := NOSHARE; RETURN
   CASE '|' : SYMB := BAR; RETURN
   CASE '<' : SYMB := M_LS; RETURN
   CASE '>' : SYMB := M_GR; RETURN
   CASE '%' : SYMB := PERCENT; RETURN
   CASE '-' : SYMB := M_NOT; RETURN

   CASE '**' : RCH()
               IF CH = '**' DO $( SYMB, CHKIND := M_POWER, EMPTY
                                RETURN $)
               SYMB := M_MULT
               RETURN

   CASE ':' : RCH()
              IF CH='=' DO $( SYMB, CHKIND := ASS, EMPTY
                             RETURN $)
              SYMB := COLON
              RETURN

```

```

CASE '-' : RCH()
           IF CH='**' LOGOR CH='>' DO
             $( SYMB, CHKIND := COND, EMPTY
                RETURN $)
           SYMB := M_MINUS
           RETURN

CASE '/' : RCH()
           IF CH='/' DO $( RCH()
                          IF CH='*N' GOTO L $) REPEAT
           SYMB := M_DIV
           RETURN

CASE '*' : SYMBP := 0

NSCH: $( RCH()
         IF CH='**' DO
           $( RCH()
             SYMBP := SYMBP + 1
             SYMBV*(SYMBP) := CH='T' -* '*T',
                              CH='S' -* '*S',
                              CH='N' -* '*N',
                              CH
                               GOTO NSCH $)
         IF CH='**' DO
           $( SYMB, CHKIND := STRINGCONST, EMPTY
              RETURN $)

         IF CH='*N' LOGOR CH=ENDOFSTREAMCH DO
           $( REPORT(5, 2, 'UNCLOSED QUOTE')
              SYMB, CHKIND := STRINGCONST, EMPTY
              RETURN $)
         SYMBP := SYMBP + 1
         SYMBV*(SYMBP) := CH
         GOTO NSCH $)

CASE '#' : IF LINEP NE 1 GOTO CONTERR
           WRITECH(OUTPUT, '*B')
CASE ENDOFSTREAMCH: SYMB := END
                   RETURN

CONTERR:
DEFAULT: REPORT(5, 4, 'CHARACTER OUT OF CONTEXT')
         GOTO L $)1

```

```
AND KIND() = VALOF
```

```
$(1 SWITCHCN CH INTO
```

```

$( CASE 'A':CASE 'B':CASE 'C':CASE 'D':CASE 'E':
   CASE 'F':CASE 'G':CASE 'H':CASE 'I':CASE 'J':
   CASE 'K':CASE 'L':CASE 'M':CASE 'N':CASE 'O':
   CASE 'P':CASE 'Q':CASE 'R':CASE 'S':CASE 'T':
   CASE 'U':CASE 'V':CASE 'W':CASE 'X':CASE 'Y':
   CASE 'Z':CASE '_':

```

```
RESULTIS CAPITAL
```

```

CASE '0':CASE '1':CASE '2':CASE '3':CASE '4':
CASE '5':CASE '6':CASE '7':CASE '8':CASE '9':
    RESULTIS DIGIT

```

```

CASE '*N':CASE '*S':CASE '*T':CASE 0:
    RESULTIS IGNORABLE

```

```

CASE ';':CASE ',':CASE '+':CASE '(':
CASE ')':CASE '=':CASE '&':CASE '$':
CASE '>':CASE '<':CASE '|':CASE '%':
CASE '~':
    RESULTIS SIMPLE

```

```

CASE '.': RESULTIS DOTK

```

```

DEFAULT: RESULTIS OTHERS $)1

```

```

AND RCH() BE

```

```

$(1 IF CH='*N' DO $( LINEP := 0; COLUMN(OUTPUT, 21) := $)
  READCH(INPUT, LVCH)
  LINEP := LINEP+1
  IF LINEP > LINET DO
    $( WRITECH(OUTPUT, '*T') // SEPARATE COLUMNS 72 AND 73
      $( WRITECH(OUTPUT, CH)
        READCH(INPUT, LVCH) $) REPEATUNTIL CH='*N'
        IF TIME_EXCEEDED DO TIMECVFL() $)
      WRITECH(OUTPUT, CH) $)1

```

```

>>> EJECT
// PAL1B

```

```

LET LOOKUPWORD() = VALOF

```

```

$(1 LET I, V2, V3, V4, V5, V6 = SYMBP, SYMBV*(2), SYMBV*(3),
  SYMBV*(4), SYMBV*(5), SYMBV*(6)

```

```

SWITCHON SYMBV*(1) INTO
$( DEFAULT: RESULTIS NAME

```

```

CASE 'A':
RESULTIS I=3 & V2='N' & V3='D' -* M_AND,
  I=3 & V2='U' & V3='G' -* M_AUG,
  NAME

```

```

CASE 'D':
RESULTIS I=3 & V2='E' & V3='F' -* DEF,
  I=5 & V2='U' & V3='M' & V4='M' & V5='Y' -* M_DUMMY,
  I=2 & V2='O' -* M_DO, NAME

```

```

CASE 'E':
RESULTIS I=2 & V2='Q' -* M_EQ,
  NAME

```

CASE 'F':  
 RESULT IS I=5 & V2='A' & V3='L' & V4='S' & V5='E' -\* M\_FALSE,  
 I=2 & V2='N' -\* LAMBDA, NAME

CASE 'G':  
 RESULT IS I=2 & V2='R' -\* M\_GR,  
 I=2 & V2='E' -\* M\_GE,  
 I=4 & V2='O' & V3='T' & V4='O' -\* M\_GOTO, NAME

CASE 'I':  
 RESULT IS I=2 & V2='F' -\* M\_IF,  
 I=4 & V2='F' & V3='S' & V4='O' -\* IFSO,  
 I=5 & V2='F' & V3='N' & V4='O' & V5='T' -\* IFNOT,  
 I=2 & V2='N' -\* IN, NAME

CASE 'L':  
 RESULT IS I=3 & V2='E' & V3='T' -\* M\_LET,  
 I=2 & V2='S' -\* M\_LS,  
 I=2 & V2='E' -\* M\_LE,  
 NAME

CASE 'N':  
 RESULT IS I=3 & V2='O' & V3='T' -\* M\_NOT,  
 I=2 & V2='E' -\* M\_NE,  
 I=3 & V2='I' & V3='L' -\* M\_NIL, NAME

CASE 'O':  
 RESULT IS I=2 & V2='R' -\* M\_LOGOR, NAME

CASE 'R':  
 RESULT IS I=3 & V2='E' & V3='C' -\* REC,  
 I=3 & V2='E' & V3='S' -\* M\_RES,  
 NAME

CASE 'T':  
 RESULT IS I=4 & V2='R' & V3='U' & V4='E' -\* M\_TRUE,  
 I=4 & V2='E' & V3='S' & V4='T' -\* M\_TEST, NAME

CASE 'V':  
 RESULT IS I=5 & V2='A' & V3='L' & V4='O' & V5='F' -\* M\_VALOF,  
 NAME

CASE 'W':  
 RESULT IS I=5 & V2='H' & V3='E' & V4='R' & V5='E' -\* WHERE,  
 I=5 & V2='H' & V3='I' & V4='L' & V5='E' -\* M\_WHILE,  
 I=6 & V2='I' & V3='T' & V4='H' & V5='I' & V6='N' -\* WITHIN,  
 NAME



```
// PAL2 LAST MODIFIED ON FRIDAY, 12 JUNE 1970
// AT 5:37:16.75 BY R MABEE
>>> FILENAME 'PAL2'
```

```
//
// *****
// * PAL2 *
// * *
// *****
//
```

```
>>> GET 'PALHD'
```

```
>>> EJECT
// PAL2A
```

```
MANIFEST $( EMPTY = 0 $)
```

```
LET CAE() = VALOF
$(1 LET A, I = 0, 0
LET DEFV = VEC (BYTEMAX/BYTESPERWORD)
LET V1 = VEC BYTEMAX
SYMBV, SYMBP := V1, 0
LINEP := 0
AETREEP := STORAGET
CHKIND, CH := EMPTY, '*N'
NAMECHAIN := 0
DUMMYN := LIST1(M_DUMMY)
NEXTSYMB()
TEST SYMB=DEF
THEN $(3 L: WHILE SYMB=DEF DO
$( NEXTSYMB()
DEFV*(I) := RDEF(0)
I := I+1 $)
UNLESS SYMB=END DO
$(4 REPORT(2, 97, '**DEF*' DEFINITION')
RCCMLOOP(1)
N: SWITCHON SYMB INTO
$( DEFAULT: NEXTSYMB()
GOTO N
CASE DEF: GOTO L
CASE END: $)4
A := LIST1(M_DUMMY)
UNTIL I=0 DO $( I := I-1
A := LIST3(DEF, DEFV*(I), A) $)3
OR $(3 P: A := LIST2(PAREN, RCCM(0) )
UNLESS SYMB=END DO
$( REPORT(2, 98, 'THE PROGRAM IS')
RCCMLOOP(0) $)3
RESULT IS A $)1
```

```

AND RCCMLOOP(N) BE
  $( $( NEXTSYMB()
    RCCM(0)
    IF SYMB=END RETURN
    IF N=1 LOGAND ( SYMB=DEF LOGOR SYMB=M_AND ) RETURN
    REPORT(2, 99, 'THE PROGRAM OR DEFINITION HAS AGAIN')
    $) REPEAT $)

```

```

AND REPORT(M, N, S) BE
  $(1 WRITES('N*T*T*****SYNTAX ERROR ')
    WRITEN(N)
    WRITES(' ... ')
    SWITCHON M INTO

    $( CASE 1: WRITES('SYNTAX ERROR IN ')
      TEST S=0
      THEN WRITES('DEFINITION')
      OR $( WRITES(S)
        WRITES(' EXPRESSION') $)
      GOTO L

    CASE 2: WRITES(S)
      WRITES(' PREMATURELY TERMINATED')
      GOTO L

    CASE 3: WRITES(S)
      WRITES(' OUT OF CONTEXT')
      GOTO L

    CASE 4: WRITES('UNMATCHED CLOSING BRACKET IN ')
    CASE 5: WRITES(S) $)

  L: WRITES('N*T*T')
  UNLESS CH = 'N' DO FOR I = 1 TO LINEP DO WRITECH(OUTPUT, '*S')
  IF N LS 100 DO COMPEROR := TRUE $)1

  >>> EJECT
  // PAL2B

```

```

LET RCCM(N) = VALOF

```

```

$(1 LET A, B, C = 0, 0, 0

```

```

  SWITCHON SYMB INTO

```

```

  $( CASE M_LET: UNLESS N=0 DO REPORT(3, 30, '**LET**')
    NEXTSYMB()
    A := RDEF(0)
    UNLESS SYMB=IN DO
      REPORT(1, 31, '**LET**')
    NEXTSYMB()

```

```

      B := RCCM(0)
      RESULTIS LIST3(M_LET, A, B)

CASE LAMBDA: UNLESS N=0 DO REPORT(3, 32, '**FN**')
      NEXTSYMB()
      $( LET V = VEC 50
        LET I = 0
        WHILE I LE 50 DO
          $( UNLESS SYMB=ERA LOGOR SYMB=NAME BREAK
            V*(I) := RBV()
            I := I+1  $)
          IF I=0 DO REPORT(1, 33, '**FN**')
          UNLESS SYMB=DOT DO REPORT(1, 34, '**FN**')
          NEXTSYMB()
          A := RCOM(0)
          WHILE I GR 0 DO
            $( I := I-1
              A := LIST3(LAMBDA, V*(I), A)  $)
          RESULTIS A  $)

CASE M_VALOF: UNLESS N LE 4 DO REPORT(3, 35, '**VALOF**')
      NEXTSYMB()
      B := RCCM(6)
      A := LIST2(M_VALOF, B)
      GOTO L

CASE M_TEST: UNLESS N LE 10 DO REPORT(3, 36, '**TEST**')
      NEXTSYMB()
      A := REXP(20)
      SWITCHON SYMB INTO
      $( CASE IFSO: NEXTSYMB()
          B := RCCM(8)
          UNLESS SYMB=IFNOT GOTO TESTERR
          NEXTSYMB()
          C := RCCM(8)
          TESTEND: A := LIST4(COND, A, B, C)
          GOTO L
        CASE IFNOT: NEXTSYMB()
          C := RCCM(8)
          UNLESS SYMB=IFSO GOTO TESTERR
          NEXTSYMB()
          B := RCCM(8)
          GOTO TESTEND
        DEFAULT:
          TESTERR: REPORT(1, 37, '**TEST**')
          GOTO L  $)

CASE M_IF:
CASE M_WHILE: $( LET OP = SYMB
                UNLESS N LE 10 DO
                  REPORT(3, 38, '**IF** OR **WHILE**')
                NEXTSYMB()
                A := REXP(20)
                TEST SYMB=M_DO
                THEN NEXTSYMB()

```

```

        OR   REPORT(5, 138,
                '**DC*' ASSUMED TO BE MISSING.')
```

B := RCOM(8)  
 TEST OP=M\_IF  
 THEN A := LIST4(COND,A,B,DUMMYN)  
 OR A := LIST3(M\_WHILE,A,B)  
 GOTO L \$)

CASE M\_GOTO: NEXTSYMB()  
 B := REXP(38)  
 A := LIST2(M\_GOTO, B)  
 GOTO L

CASE M\_RES: NEXTSYMB()  
 B := REXP(14)  
 A := LIST2(M\_RES, B)  
 GOTO L

CASE M\_DUMMY: NEXTSYMB()  
 A := DUMMYN  
 GOTO L

DEFAULT: A := REXP(N)  
 UNLESS SYMB=ASS GOTO L  
 NEXTSYMB()  
 B := REXP(14)  
 A := LIST3(ASS, A, B)  
 GOTO L \$)

L: SWITCHCN SYMB INTO

\$( CASE WHERE: IF N GR 2 RESULTIS A  
 NEXTSYMB()  
 B := RBDEF(0)  
 RESULTIS LIST3(M\_LET, B, A)

CASE SEQ: IF N GR 6 RESULTIS A  
 NEXTSYMB()  
 B := RCOM(6)  
 A := LIST3(SEQ, A, B)  
 GOTO L

CASE COLON: UNLESS HI\*(A)=NAME LOGAND N LE 8 DO  
 REPORT(5, 39, 'SYNTAX ERROR IN LABEL')  
 NEXTSYMB()  
 B := RCOM(8)  
 A := LIST4(COLON, A, B, 0)  
 GOTO L

DEFAULT: RESULTIS A \$)1

```

>>> EJECT
// PAL2C
```

```
LET REXP(N) = VALOF
```

```
$(1 LET A, B, C = 0, 0, 0
```

```
SWITCHON SYMB INTO
```

```
$(2 CASE M_NOT: UNLESS N LE 24 DO REPORT(3, 51, '**NOT**')
NEXTSYMB()
A := REXP(26)
A := LIST2(M_NOT, A)
GOTO L
```

```
CASE M_PLUS:
CASE M_MINUS: $( LET OP = SYMB
NEXTSYMB()
UNLESS N LE 30 DO
REPORT(3, 52, '**+*' OR '**-*')
A := REXP(32)
A := LIST2(OP=M_PLUS -* M_NEG, M_NEG, A)
GOTO L $)
```

```
CASE NOSHARE: UNLESS N LE 36 DO REPORT(3, 53, '**$**')
NEXTSYMB()
B := REXP(38)
A := LIST2(NOSHARE, B)
GOTO L
```

```
CASE M_NIL:
CASE M_TRUE:
CASE M_FALSE: A := LIST1(SYMB)
NEXTSYMB()
GOTO APPLY
```

```
CASE NUMBER:
CASE STRINGCONST: A := RCNS()
NEXTSYMB()
GOTO APPLY
```

```
CASE NAME: A := RDNAM()
APPLY: B := RARG()
IF B = 0 GOTO L
A := LIST3(M_APPLY, A, B)
GOTO APPLY
```

```
DEFAULT: A := RDBEXP()
IF A=0 DO
$( TEST SYMB=END
THEN REPORT(2, 55, 'SOURCE PROGRAM')
OR REPORT(3, 56, 'SYMBOL')
RESULTIS 0 $)
IF N LE 8 DO A := H2*(A)
GOTO APPLY $)2
```

```
L: SWITCHON SYMB INTO
```

```

$( DEFAULT: RESULTIS A

CASE COMMA: IF N GR 14 RESULTIS A
$( LET I = 1
LET V = VEC 500
WHILE SYMB = COMMA DO
$( NEXTSYMB()
V*(I) := REXP(16)
I := I + 1 $)
B := A
A := NEWVEC(I + 1)
A*(0), A*(1), A*(2) := COMMA, I, B
FOR J = 1 TO I - 1 DO A*(J + 2) := V*(J) $)
GOTO L

CASE M_AUG: IF N GR 16 RESULTIS A
NEXTSYMB()
B := REXP(18)
A := LIST3(M_AUG, A, B)
GOTO L

CASE COND: IF N GR 18 RESULTIS A
NEXTSYMB()
B := REXP(18)
UNLESS SYMB=BAR DO REPORT(1, 57, '**->**')
NEXTSYMB()
C := REXP(18)
A := LIST4(COND, A, B, C)
GOTO L

CASE M_LOGOR: IF N GR 20 RESULTIS A
NEXTSYMB()
B := REXP(22)
A := LIST3(M_LOGOR, A, B)
GOTO L

CASE M_LOGAND: IF N GR 22 RESULTIS A
NEXTSYMB()
B := REXP(24)
A := LIST3(M_LOGAND, A, B)
GOTO L

CASE VALDEF: REPORT(5, 157,
**'=**' USED OUT OF CONTEXT: **EQ** ASSUMED**)
SYMB := M_EQ

CASE M_GE:
CASE M_NE:
CASE M_LE:
CASE M_EQ:
CASE M_LS:
CASE M_GR: IF N GR 26 RESULTIS A
$( LET OP = SYMB
NEXTSYMB()
B := REXP(30)
A := LIST3(OP, A, B)

```

GOTO L \$)

```

CASE M_PLUS:
CASE M_MINUS: $( LET OP = SYMB
                 IF N GR 30 RESULTIS A
                 NEXTSYMB()
                 B := REXP(32)
                 A := LIST3(OP, A, B)
                 GOTO L $)

```

```

CASE M_MULT:
CASE M_DIV:   IF N GR 32 RESULTIS A
CASE M_POWER: IF N GR 36 RESULTIS A
              $( LET OP = SYMB
                NEXTSYMB()
                B := REXP(34)
                A := LIST3(OP, A, B)
                GOTO L $)

```

```

CASE PERCENT: IF N GR 36 RESULTIS A
              NEXTSYMB()
              UNLESS SYMB=NAME DO REPORT(3, 58, '**%*')
              B := RDNAM()
              C := REXP(38)
              A := LIST4(COMMA, 2, A, C)
              A := LIST3(M_APPLY, B, A)
              GOTO L $)1

```

```

>>> EJECT
// PAL2D

```

LET RBDEF(N) = VALGF

```

$(1 LET A=0
   SWITCHGN SYMB INTO

```

```

$( CASE NAME: $(2 LET B=0
                  A := RDNAM()
                  IF SYMB=COMMA DO
                    $( A := RDNAMLIST(A)
                      UNLESS SYMB=VALDEF DO REPORT(1, 10, 0)
                      NEXTSYMB()
                      B := RCOM(0)
                      RESULTIS LIST3(VALDEF, A, B) $)

```

```

                  IF SYMB=VALDEF DO
                    $( NEXTSYMB()
                      B := RCCM(0)
                      RESULTIS LIST3(VALDEF, A, B) $)

```

```

$( LET V = VEC 10
   LET I = 0
   WHILE I LE 10 DO
     $( UNLESS SYMB=BRA LOGOR SYMB=NAME BREAK

```

```

        V*(I) := RBV()
        I := I + 1 $)
    UNLESS I NE 0 LOGAND SYMB=VALDEF DO
        REPORT(1, 11, 0)
    NEXTSYMB()
    B := RCOM(0)
    WHILE I GR 0 DO $( I := I - 1
                        B := LIST3(LAMBDA, V*(I), B) $)
    RESULTIS LIST3(VALDEF, A, B) $)2

CASE BRA: NEXTSYMB()
    A := RDEF(0)
    UNLESS SYMB=KET DO REPORT(4, 12, 'DEFINITION')
    NEXTSYMB()
    RESULTIS A

CASE REC: NEXTSYMB()
    UNLESS N EQ 0 DO
        $( REPORT(5, 112, 'REDUNDANT *'REC*' IGNORED')
           RESULTIS RBDEF (2) $)
    A := RBDEF(2)
    RESULTIS LIST2(REC, A)

DEFAULT: REPORT(1, 13, 0)
    RESULTIS 0 $)1

```

AND RDEF(N) = VALDF

```

$(1 LET A = RBDEF(0)
   LET B = 0

```

L: SWITCHCN SYMB INTO

```

$( DEFAULT: RESULTIS A

```

```

CASE M_AND: IF A = 0 DO
    REPORT(5,15,'DEFINITION MISSING BEFORE *'AND*')
    IF N GE 6 RESULTIS A
    $( LET I = 1
       LET V = VEC 100
       WHILE SYMB = M_AND DO
           $( NEXTSYMB()
              V*(I) := RBDEF(0)
              I := I + 1 $)
       B := A
       A := NEWVEC(I + 1)
       A*(0), A*(1), A*(2) := M_AND, I, B
       FOR J = 1 TO I - 1 DO A*(J + 2) := V*(J) $)
    GOTO L

```

```

CASE WITHIN: IF A=0 DO REPORT(5, 16,
    'DEFINITION MISSING BEFORE *'WITHIN*')
    IF N GE 3 RESULTIS A
    NEXTSYMB()

```



```

B := RDEF(0)
A := LIST3(WITHIN, A, B)
GOTO L    $)1

```

AND RBV() = VALOF

```

$(1 LET A=0
  IF SYMB=NAME RESULTIS RDNAM()
  NEXTSYMB()
  IF SYMB=KET DO $( NEXTSYMB()
                    RESULTIS LIST1(MPT) $)
  A := RDNAMLIST(0)
  UNLESS SYMB=KET DO
    REPORT(4, 17, 'BV PART')
  NEXTSYMB()
  RESULTIS A    $)1

```

AND RDNAMLIST(N) = VALOF

```

$( LET A, B, I = 0, N, 1
  LET V = VEC 100
  IF N = 0 DO $( UNLESS SYMB=NAME DO
                REPORT(5, 20, 'A NAME IS MISSING')
                B := RDNAM() $)
  UNLESS SYMB = COMMA RESULTIS B
  WHILE SYMB = COMMA DO
    $( NEXTSYMB()
      UNLESS SYMB=NAME DO
        REPORT(5, 21, 'A NAME IS MISSING')
        V*(I) := RDNAM()
        I := I + 1 $)
  A := NEWVEC(I + 1)
  A*(0), A*(1), A*(2) := COMMA, I, B
  FOR J = 1 TO I - 1 DO A*(J + 2) := V*(J)
  RESULTIS A    $)

```

AND RDNAM() = VALCF

```

$(1 LET S = VEC (BYTEMAX/BYTESPERWORD)
  LET L, A, B = NAMECHAIN, 0, SYMB
  LET N = SYMBP/BYTESPERWORD + 1
          // THE LENGTH OF THE STRING IN WORDS
  IF N GR 5 DO REPORT(5, 23, 'NAME TOO LONG')
  SYMBV*(0) := SYMBP
  PACKSTRING(SYMBV, S)
  NEXTSYMB()
  UNTIL L = 0 DO
    $(2 LET V = H3*(L)
      IF S*(0)=V*(0)
      DO $(3 IF N=1 RESULTIS L
            IF S*(1)=V*(1)
            DC $( IF N=2 RESULTIS L

```

```

                IF S*(2)=V*(2)
                DO $( IF N=3 RESULTIS L
                    IF S*(3)=V*(3)
                    DO $( IF N=4 RESULTIS L
                        IF S*(4)=V*(4)
                        CC RESULTIS L $)3
                L := H2*(L) $)2
A := NEWVEC(N-1)
NAMECHAIN := LIST3(B, NAMECHAIN, A)
FOR I = 0 TO N-1 DO A*(I) := S*(I)
RESULTIS NAMECHAIN $)1

```

AND RDNS() = VALOF

```

$( LET A = 0
  LET N = SYMBP/BYTESPERWORD + 1
  LET S = VEC 150
  SYMBV*(0) := SYMBP
  PACKSTRING(SYMBV, S)
  A := NEWVEC(N-1)
  FOR I = 0 TO N-1 DO A*(I) := S*(I)
  RESULTIS LIST2(SYMB, A) $)

```

AND RARG() = VALOF

```

$(1 LET A = 0
  SWITCHON SYMB INTO

  $( DEFAULT: RESULTIS RDBEXP()

  CASE M_NIL:
  CASE M_TRUE:
  CASE M_FALSE: A := LIST1(SYMB)
                NEXTSYME()
                RESULTIS A

  CASE NUMBER:
  CASE STRINGCONST: A := RDNS()
                    NEXTSYMB()
                    RESULTIS A

  CASE NAME: RESULTIS RDNAM() $)1

```

AND RCBEXP() = VALOF

```

$(1 LET A=0
  UNLESS SYMB=BRA RESULTIS C
  NEXTSYMB()
  A := RCOM(C)
  IF A=0 DO REPORT(5, 25,
    'EXPRESSION MISSING WITHIN BRACKETS')
  UNLESS SYMB=KET DO

```

```

REPORT(4, 26, 'EXPRESSION')
NEXTSYMB()
RESULTIS LIST2(PAREN, A)          $)1

```

```

>>> EJECT
// PAL2E

```

```

LET PLIST(X, N, D) BE

```

```

$(1 LET SIZE, S = 0, 0
  IF X=0 DO $( WRITES( 'NIL' )
              RETURN $)
  IF X LE 100 DO $( WRITEN(X)
                  RETURN $)
  IF H1*(X)=NUMBER DO $( WRITES( '** NUMBER ' )
                        WRITES(H2*(X))
                        RETURN $)
  IF H1*(X)=NAME DO $( WRITES( '** NAME ' )
                       WRITEN(H3*(X))
                       WRITES( ' ' )
                       WRITES(H3*(X))
                       RETURN $)
  IF H1*(X)=STRINGCONST DO $( WRITES( '** STRINGCONST ' )
                              WRITES(H2*(X))
                              RETURN $)
  IF N=D DO $( WRITES( 'ETC' )
              RETURN $)
  NODETYPE(X, LV SIZE, LV S)
  WRITES(S)
  FOR I = 2 TO SIZE DO
    $( WRITECH(OUTPUT, '*N*')
      FOR I = 0 TO N DO WRITES( '| ' )
      PLIST(H1*(X+I-1), N+1, D) $)
  RETURN $)1

```

```

AND NODETYPE(X, N, S) BE

```

```

$(1 SWITCHON H1*(X) INTO

```

```

$( DEFAULT:          RV N, RV S := 0, 'UNKNOWN OPERATOR'; RETURN

CASE PAREN:         RV N, RV S := 2, 'PAREN';      RETURN
CASE DEF:           RV N, RV S := 3, 'DEF';        RETURN
CASE M_LET:         RV N, RV S := 3, 'LET';        RETURN
CASE COLON:         RV N, RV S := 3, 'COLON';      RETURN
CASE SEQ:           RV N, RV S := 3, 'SEQ';        RETURN
CASE M_GOTO:        RV N, RV S := 2, 'GOTO';        RETURN
CASE M_VALOF:       RV N, RV S := 2, 'VALOF';      RETURN
CASE M_RES:         RV N, RV S := 2, 'RES';        RETURN
CASE LAMBDA:        RV N, RV S := 3, 'LAMBDA';     RETURN
CASE COND:          RV N, RV S := 4, 'COND';       RETURN
CASE M_WHILE:       RV N, RV S := 3, 'WHILE';     RETURN
CASE ASS:           RV N, RV S := 3, 'ASS';        RETURN
CASE COMMA:         RV N, RV S := H2*(X)+2, 'COMMA'; RETURN
CASE M_AUG:         RV N, RV S := 3, 'AUG';        RETURN

```

```

CASE M_LOGOR:      RV N, RV S := 3, 'LOGOR';   RETURN
CASE M_LOGAND:    RV N, RV S := 3, 'LOGAND';  RETURN
CASE M_NOT:       RV N, RV S := 2, 'NOT';     RETURN
CASE M_EQ:        RV N, RV S := 3, 'EQ';      RETURN
CASE M_LS:        RV N, RV S := 3, 'LS';      RETURN
CASE M_GR:        RV N, RV S := 3, 'GR';      RETURN
CASE M_GE:        RV N, RV S := 3, 'GE';      RETURN
CASE M_LE:        RV N, RV S := 3, 'LE';      RETURN
CASE M_NE:        RV N, RV S := 3, 'NE';      RETURN
CASE M_PLUS:      RV N, RV S := 3, 'PLUS';    RETURN
CASE M_MINUS:     RV N, RV S := 3, 'MINUS';   RETURN
CASE M_POS:       RV N, RV S := 2, 'POS';     RETURN
CASE M_NEG:       RV N, RV S := 2, 'NEG';     RETURN
CASE M_MULT:      RV N, RV S := 3, 'MULT';    RETURN
CASE M_DIV:       RV N, RV S := 3, 'DIV';     RETURN
CASE M_POWER:     RV N, RV S := 3, 'POWER';   RETURN
CASE M_APPLY:     RV N, RV S := 3, 'APPLY';   RETURN
CASE M_DUMMY:     RV N, RV S := 1, 'DUMMY';   RETURN
CASE M_NOSHARE:   RV N, RV S := 2, 'NOSHARE'; RETURN
CASE M_TRUE:      RV N, RV S := 1, 'TRUE';    RETURN
CASE M_FALSE:     RV N, RV S := 1, 'FALSE';   RETURN
CASE M_NIL:       RV N, RV S := 1, 'NIL';     RETURN
CASE MPT:         RV N, RV S := 1, '()';      RETURN
CASE M_AND:       RV N, RV S := H2*(X)+2, 'AND'; RETURN
CASE WITHIN:     RV N, RV S := 3, 'WITHIN';   RETURN
CASE REC:        RV N, RV S := 2, 'REC';     RETURN
CASE VALDEF:     RV N, RV S := 3, 'VALDEF';   RETURN  $)1

```

```

>>> EJECT
// PAL2F

```

```
LET NEWVEC( N ) = VALOF
```

```

$(1 AETREEP := AETREEP - N - 1
  IF CODEFILEP GE AETREEP DO
    $( WRITES(*N*N*N*TAE TREE EXCEEDS AVAILABLE SPACE. *)
      WRITES('COMPILATION ABORTED.*N')
      COMPERROR := TRUE
      LONGJUMP(EOP, EOPLEVEL)  $)
  RESULTIS AETREEP  $)1

```

```
AND LIST1(A) = VALOF
```

```

$(1 LET V = NEWVEC(0)
  V*(0) := A
  RESULTIS V  $)1

```

```
AND LIST2(A, B) = VALOF
```

```

$(1 LET V = NEWVEC(1)
  V*(0), V*(1) := A, B
  RESULTIS V  $)1

```

AND LIST3(A, B, C) = VALCF

```
$(1 LET V = NEWVEC(2)
  V*(0), V*(1), V*(2) := A, B, C
  RESULTIS V      $)1
```

AND LIST4(A, B, C, D) = VALCF

```
$(1 LET V = NEWVEC(3)
  V*(0), V*(1), V*(2), V*(3) := A, B, C, D
  RESULTIS V      $)1
```

```
//      PAL3      LAST MODIFIED ON FRIDAY, 12 JUNE 1970
//      AT 5:37:22.18 BY R MABEE
>>> FILENAME 'PAL3'
```

```
//
//      *****
//      *           *
//      *   PAL3   *
//      *           *
//      *****
//
```

```
>>> GET 'PALHD'
```

```
>>> EJECT
// PAL3A
```

```
LET TRANS(X, MODE) BE
```

```
$(1 IF TIME_EXCEEDED DO TIMECVFL()
```

```
IF X=0 DO $( WRITES( '*N*N*T*****EXPRESSION MISSING*N' )
COMPERROR := TRUE
OUTP(M_NIL)
UPSSP(1)
RETURN $)
```

```
$( LET OP = H1*(X)
```

```
SWITCHCN OP INTO
```

```
$( CASE M_LET: $( LET L = NEXTPARAM()
LET N = NEXTPARAM()
TRANSRHS(H2*(X))
OUTP(M_BLOCKLINK); OUTP(L)
IF SSP=MSP DO MSP := SSP+1
TRANSSCOPE(X, N, MODE)
COMPLAB(L)
RETURN $)
```

```
CASE DEF: TRANSRHS(H2*(X))
C_DECLNAMES(H2*(X))
TRANSLABELS(H3*(X))
TRANS(H3*(X), VAL)
RETURN
```

```
CASE M_MULT:CASE M_DIV:CASE M_PLUS:CASE M_MINUS:CASE M_POWER:
CASE M_EQ:CASE M_LS:CASE M_GR:
CASE M_GE: CASE M_LE: CASE M_NE:
CASE M_LOGAND:CASE M_LOGOR:
TRANS(H3*(X), VAL)
TRANS(H2*(X), VAL)
```

```

      OUTOP(OP)
      SSP := SSP-1
      IF MODE=REF DO OUTOP(M_FORMLVALUE)
      RETURN

CASE M_AUG: TRANS(H3*(X), REF)
            TRANS(H2*(X), VAL)
            OUTOP(M_AUG)
            SSP := SSP-1
            IF MODE=REF DO OUTOP(M_FORMLVALUE)
            RETURN

CASE M_APPLY: TRANS(H3*(X), REF)
              TRANS(H2*(X), REF)
              OUTOP(M_APPLY)
              SSP := SSP-1
              IF MODE=VAL DO OUTOP(M_FORMRVALUE)
              RETURN

CASE M_POS:CASE M_NEG:CASE M_NOT:
            TRANS(H2*(X), VAL)
            OUTOP(OP)
            IF MODE=REF DO OUTOP(M_FORMLVALUE)
            RETURN

CASE NOSHARE:      TRANS(H2*(X), VAL)
                  IF MODE=REF DO OUTOP(M_FORMLVALUE)
                  RETURN

CASE COMMA: $( LET R(X) BE $( TRANS(X, REF) $)
              MAPB(R, X)
              OUTOP(M_TUPLE); OUTN(C_LENGTH(X))
              SSP := SSP - C_LENGTH(X) + 1
              IF MODE=REF DO OUTOP(M_FORMLVALUE)
              RETURN $)

CASE LAMBDA: $( LET L, M = NEXTPARAM(), NEXTPARAM()
               LET N = NEXTPARAM()
               OUTOP(M_FORMCLOSURE); OUTP(L)
               UPSSP(1)
               OUTOP(M_JUMP); OUTP(M)
               //FOR THE JUMP ROUND THE BODY
               COMPLAB(L)
               TRANSSCOPE(X, N, REF)
               COMPLAB(M)
               IF MODE=REF DO OUTOP(M_FORMLVALUE)
               RETURN $)

CASE COLON: IF H4*(X)=0 DO
            $( WRITES('*N*N*T*****LABEL *')
              WRITES(H3*(H2*(X)))
              WRITES(' IMPROPERLY USED*N')
              COMPERROR := TRUE $)
            COMPLAB(H4*(X))
            TRANS(H3*(X), MODE)

```

RETURN

```
CASE SEQ: TRANS(H2*(X), VAL)
  OUTOP(M_LOSE1)
  SSP := SSP-1
  TRANS(H3*(X), MODE)
  RETURN
```

```
CASE M_VALOF: $( LET L = NEXTPARAM()
  LET N = NEXTPARAM()
  CUTOP(M_RESLINK); OUTP(L)
  SSP := SSP+1
  IF SSP GE MSP DO MSP := SSP+1
  $( LET A, B = SSP, MSP
  SSP, MSP := 0, 1
  OUTOP(M_SAVE); OUTP(N)
  OUTOP(M_TESTEMPTY)
  OUTOP(JJ)
  OUTOP(M_FORMLVALUE)
  OUTOP(M_DECLNAME)
  OUTNAME(LIST3(NAME, 0, '***RES**'))
  TRANSLABELS(H2*(X))
  TRANS(H2*(X), REF)
  OUTOP(M_RETURN)
  UNLESS SSP=1 DO
    WRITES('*N*N*T***** SSP ERROR*N')
  OUTPSOP(N, EQU, MSP)
  SSP, MSP := A, B  $)
  COMPLAB(L)
  IF MODE=VAL DO OUTOP(M_FORMRVALUE)
  RETURN  $)
```

```
CASE M_RES: TRANS(H2*(X), REF)
  OUTOP(M_RES)
  RETURN
```

```
CASE M_GOTO: TRANS(H2*(X), VAL)
  CUTOP(M_GOTO)
  RETURN
```

```
CASE COND: $( LET L, M = NEXTPARAM(), NEXTPARAM()
  TRANS(H2*(X), VAL)
  CUTOP(M_JUMPF); OUTP(L)
  SSP := SSP-1
  TRANS(H3*(X), MODE)
  OUTOP(M_JUMP); OUTP(M)
  COMPLAB(L)
  SSP := SSP-1
  TRANS(H4*(X), MODE)
  COMPLAB(M)
  RETURN  $)
```

```
CASE M_WHILE: $( LET L, M = NEXTPARAM(), NEXTPARAM()
  COMPLAB(M)
  TRANS(H2*(X), VAL)
```



```

        OUTOP(M_JUMPF); OUTP(L)
        SSP := SSP - 1
        TRANS(H3*(X), VAL)
        OUTOP(M_LOSE1)
        OUTOP(M_JUMP); OUTP(M)
        COMPLAB(L)
        OUTOP(M_DUMMY)
        IF MODE=REF DO OUTOP(M_FORMLVALUE)
        RETURN $)

CASE ASS: TRANS(H2*(X), REF)
        TRANS(H3*(X), VAL)
        OUTOP(M_UPDATE); OUTN(C_LENGTH(H2*(X)))
        SSP := SSP-1
        IF MODE=REF DO OUTOP(M_FORMLVALUE)
        RETURN

CASE PAREN: TRANS(LABELS(H2*(X))
        TRANS(H2*(X), MODE)
        RETURN

CASE M_NIL:
CASE M_DUMMY:
CASE M_TRUE:
CASE M_FALSE: OUTOP(OP)
        UPSSP(1)
        IF MODE=REF DO OUTOP(M_FORMLVALUE)
        RETURN

CASE NAME: OUTOP(MODE=VAL -* M_LOADR, M_LOADL); OUTNAME(X)
        UPSSP(1)
        RETURN

CASE NUMBER: OUTOP(M_LOADN); OUTNUMBER(X)
        UPSSP(1)
        IF MODE=REF DO OUTOP(M_FORMLVALUE)
        RETURN

CASE STRINGCONST: OUTOP(M_LOADS); OUTSTRING(X)
        UPSSP(1)
        IF MODE=REF DO OUTOP(M_FORMLVALUE)
        RETURN $)1

AND FINDLABELS(X) = VALGF

$(1 IF X=0 RESULTIS 0
    SWITCHON H1*(X) INTO

$( DEFAULT: RESULTIS 0

CASE CCLCN: $( LET L = NEXTPARAM()
        H4*(X) := L
        OUTOP(M_DECLLABEL); OUTNAME(H2*(X)); OUTP(L)
        RESULTIS 1 + FINDLABELS(H3*(X)) $)

```

```

CASE PAREN:      RESULTIS FINDLABELS(H2*(X))
CASE COND:       RESULTIS FINDLABELS(H3*(X))+FINDLABELS(H4*(X))
CASE M_WHILE:    RESULTIS FINDLABELS(H3*(X))
CASE SEQ:        RESULTIS FINDLABELS(H2*(X))+FINDLABELS(H3*(X)) $)1

```

AND TRANSLABELS(X) BE

```

$( LET N = FINDLABELS(X)
  IF N NE 0 DO $( OUTOP(M_SETLABES); OUTN(N) ; $) ; $)

```

```

>>> EJECT
// PAL3B

```

LET TRANSRHS(X) BE

```

$(1 IF X=0 RETURN
  SWITCHON H1*(X) INTO
  $( CASE M_AND: MAPB(TRANSRHS, X)
     OUTOP(M_TUPLE); OUTN(C_LENGTH(X))
     SSP := SSP - C_LENGTH(X) + 1
     OUTOP(M_FORMLVALUE)
     RETURN

```

```

CASE VALDEF: TRANS(H3*(X), REF)
  RETURN

```

```

CASE REC: OUTOP(M_LOADE)
  UPSSP(1)
  DECLGUESSES(H2*(X))
  TRANSRHS(H2*(X))
  C_INITNAMES(H2*(X))
  LOADDEFINEE(H2*(X))
  OUTOP(M_RESTOREE1)
  SSP := SSP-1
  RETURN

```

```

CASE WITHIN: $( LET L = NEXTPARAM()
  LET N = NEXTPARAM()
  TRANSRHS(H2*(X))
  OUTOP(M_BLOCKLINK); OUTP(L)
  IF SSP=MSP DO MSP := SSP+1
  $( LET A, B = SSP, MSP
     SSP, MSP := 1, 1
     OUTOP(M_SAVE); OUTP(N)
     C_DECLNAMES(H2*(X))
     TRANSRHS(H3*(X))
     OUTOP(M_RETURN)
     UNLESS SSP=1 DO
       WRITES('*N*N*T***** SSP ERROR*N')
     OUTPSOP(N, EQU, MSP)

```

```

        SSP, MSP := A, B    $)
    COMPLAB(L)    $)1

```

```

AND C_DECLNAMES(X) BE

```

```

    $(1 IF X=0 RETURN

```

```

        SWITCHON H1*(X) INTO

```

```

        $( CASE NAME: OUTOP(M_DECLNAME); OUTNAME(X)
            SSP := SSP-1
            RETURN

```

```

        CASE COMMA: OUTOP(M_DECLNAMES); OUTN(C_LENGTH(X))
            SSP := SSP-1
            MAPF(OUTNAME, X)
            RETURN

```

```

        CASE M_AND: OUTOP(M_MEMBERS); OUTN(C_LENGTH(X))
            UPSSP(C_LENGTH(X)-1)
            MAPF(C_DECLNAMES, X)
            RETURN

```

```

        CASE REC:
        CASE VALDEF: C_DECLNAMES(H2*(X))
            RETURN

```

```

        CASE WITHIN: C_DECLNAMES(H3*(X))
            RETURN

```

```

        CASE MPT: OUTOP(M_TESTEMPTY)
            SSP := SSP-1
            RETURN

```

```

        $)1

```

```

AND LOADDEFINEE(X) BE

```

```

    $(1 IF X=0 RETURN

```

```

        SWITCHON H1*(X) INTO

```

```

        $( CASE NAME: OUTOP(M_LOADR); OUTNAME(X)
            UPSSP(1)
            OUTOP(M_FORMLVALUE)
            RETURN

```

```

        CASE M_AND:
        CASE COMMA: MAPB(LOADDEFINEE, X)
            OUTOP(M_TUPLE); OUTN(C_LENGTH(X))
            SSP := SSP - C_LENGTH(X) + 1
            OUTOP(M_FORMLVALUE)
            RETURN

```

```

        CASE REC:
        CASE VALDEF: LCADDEFINEE(H2*(X))
            RETURN

```

```

        CASE WITHIN: LOADDEFINEE(H3*(X))
            RETURN    $)1

```

AND DECLGUESSES(X) BE

```
$(1 IF X=0 RETURN
  SWITCHON H1*(X) INTO
  $( CASE NAME: OUTOP(M_LOADGUESS)
      IF SSP=MSP DO MSP := SSP+1
      OUTOP(M_DECLNAME); OUTNAME(X)
      RETURN
```

```
  CASE M_AND:
  CASE COMMA: MAPF(DECLGUESSES, X)
      RETURN
```

```
  CASE REC:
  CASE VALDEF: DECLGUESSES(H2*(X))
      RETURN
```

```
  CASE WITHIN: DECLGUESSES(H3*(X))
      RETURN $)1
```

AND C\_INITNAMES(X) BE

```
$(1 IF X=0 RETURN
  SWITCHON H1*(X) INTO
  $( CASE NAME: OUTOP(M_INITNAME); OUTNAME(X)
      SSP := SSP-1
      RETURN
```

```
  CASE M_AND: OUTOP(M_MEMBERS); OUTN(C_LENGTH(X))
      UPSSP(C_LENGTH(X)-1)
      MAPF(C_INITNAMES, X)
      RETURN
```

```
  CASE COMMA: OUTOP(M_INITNAMES); OUTN(C_LENGTH(X))
      SSP := SSP-1
      MAPF(OUTNAME, X)
      RETURN
```

```
  CASE REC:
  CASE VALDEF: C_INITNAMES(H2*(X))
      RETURN
```

```
  CASE WITHIN: C_INITNAMES(H3*(X))
      RETURN $)1
```

AND TRANSSCOPE(X, N, MCDE) BE

```
$( LET A,B = SSP, MSP
  SSP, MSP := 1,1
  OUTOP(M_SAVE); OUTP(N)
  C_DECLNAMES(H2*(X))
  TRANSLABELS(H3*(X))
  TRANS(H3*(X), MCDE)
```

```

CUTOP(M_RETURN)
UNLESS SSP=1 DO WRITES('*N*N*T***** SSP ERROR*N*')
OUTPSOP(N, EQU, MSP)
SSP, MSP := A,B  $)

```

```

>>> EJECT
// PAL3C

```

```

LET MAPF(R, X) BE
$( LET J = H2*(X)
  FOR I = 1 TO J DO R(X*(I + 1))  $)

```

```

AND MAPB(R, X) BE
$( LET J = H2*(X)
  FOR I = 1 TO J DO R(X*(J - I + 2))  $)

```

```

AND C_LENGTH(X) = H1*(X)=M_AND LOGOR H1*(X)=COMMA -* H2*(X), 1

```

```

AND NEXTPARAM() = VALOF
$( PARAMNUMBER := PARAMNUMBER + 1
  RESULTIS PARAMNUMBER  $)

```

```

AND UPSSP(X) BE
$( SSP := SSP + X
  IF SSP GR MSP DO MSP := SSP  $)

```

```

>>> EJECT
// PAL3D

```

```

LET COMPLAB(N) BE
$( OUT2(LAB, N)
  UNLESS LISTING RETURN
  WRITES('*N*N*L*')
  WRITEN(N)  $)

```

```

AND OUTOP(OP) BE
$(1 OUT1(OP)
  UNLESS LISTING RETURN
  $( LET S = VALOF
    $(1 SWITCHON OP INTO
      $( DEFAULT:
          RESULTIS 'ERROR'
        CASE M_RESTOREE1:
          RESULTIS 'RESTOREE1'
        CASE M_FORMRVALUE:
          RESULTIS 'FORMRVALUE'
        CASE M_FORMLVALUE:
          RESULTIS 'FORMLVALUE'
        CASE M_TUPLE:
          RESULTIS 'TUPLE'

```

CASE M_MEMBERS:	RESULTIS 'MEMBERS'
CASE M_LOADGUESS:	RESULTIS 'LOADGUESS'
CASE M_TRUE:	RESULTIS 'TRUE'
CASE M_FALSE:	RESULTIS 'FALSE'
CASE M_NIL:	RESULTIS 'NIL'
CASE M_DUMMY:	RESULTIS 'DUMMY'
CASE JJ:	RESULTIS 'LOADJ'
CASE M_LOSE1:	RESULTIS 'LOSE1'
CASE M_MULT:	RESULTIS 'MULT'
CASE M_DIV:	RESULTIS 'DIV'
CASE M_POWER:	RESULTIS 'POWER'
CASE M_PLUS:	RESULTIS 'PLUS'
CASE M_MINUS:	RESULTIS 'MINUS'
CASE M_POS:	RESULTIS 'POS'
CASE M_NEG:	RESULTIS 'NEG'
CASE M_EQ:	RESULTIS 'EQ'
CASE M_LS:	RESULTIS 'LS'
CASE M_GR:	RESULTIS 'GR'
CASE M_LE:	RESULTIS 'LE'
CASE M_GE:	RESULTIS 'GE'
CASE M_NE:	RESULTIS 'NE'
CASE M_LOGAND:	RESULTIS 'LOGAND'
CASE M_LOGOR:	RESULTIS 'LOGOR'
CASE M_AUG:	RESULTIS 'AUG'
CASE M_APPLY:	RESULTIS 'APPLY'
CASE M_SAVE:	RESULTIS 'SAVE'
CASE M_NOT:	RESULTIS 'NOT'
CASE M_GOTO:	RESULTIS 'GOTO'
CASE M_RES:	RESULTIS 'RESULT'
CASE M_UPDATE:	RESULTIS 'UPDATE'
CASE M_RETURN:	RESULTIS 'RETURN'
CASE M_TESTEMPTY:	RESULTIS 'TESTEMPTY'
CASE M_LOADR:	RESULTIS 'LOADR'
CASE M_LOADL:	RESULTIS 'LOADL'
CASE M_LOADN:	RESULTIS 'LOADN'
CASE M_LOADS:	RESULTIS 'LOADS'
CASE M_LOADE:	RESULTIS 'LOADE'
CASE M_DECLNAME:	RESULTIS 'DECLNAME'
CASE M_DECLNAMES:	RESULTIS 'DECLNAMES'
CASE M_INITNAME:	RESULTIS 'INITNAME'
CASE M_INITNAMES:	RESULTIS 'INITNAMES'
CASE M_SETLABES:	RESULTIS 'SETLABES'
CASE M_FORMCLOSURE:	RESULTIS 'FORMCLOSUR'
CASE M_JUMPF:	RESULTIS 'JUMPF'
CASE M_JUMP:	RESULTIS 'JUMP'
CASE M_BLOCKLINK:	RESULTIS 'BLOCKLINK'
CASE M_RESLINK:	RESULTIS 'RESLINK'
CASE M_SETUP:	RESULTIS 'SETUP'
CASE M_DECLLABEL:	RESULTIS 'DECLLABEL' \$)1

WRITES('\*N\*T')  
WRITES(S) \$)1

AND OUTN(N) BE

```
$( OUT2(INTEGER, N)
  UNLESS LISTING RETURN
  WRITECH(OUTPUT, '*T')
  WRITEN(N) $)
```

AND OUTP(N) BE

```
$( OUT2(PARAM, N)
  UNLESS LISTING RETURN
  WRITES('*TL')
  WRITEN(N) $)
```

AND OUTNAME(N) BE

```
$( LET V = VEC BYTEMAX
  UNPACKSTRING(H3*(N), V)
  OUT1(NAME)
  FOR I = 0 TO V*(0) DO OUT1(V*(I))
  UNLESS LISTING RETURN
  WRITECH(OUTPUT, '*T')
  FOR I = 1 TO V*(0) DO WRITECH(OUTPUT, V*(I))
  $)
```

AND OUTNUMBER(N) BE

```
$( LET V = VEC BYTEMAX
  UNPACKSTRING(H2*(N), V)
  OUT1(NUMBER)
  FOR I = 0 TO V*(0) DO OUT1(V*(I))
  UNLESS LISTING RETURN
  WRITECH(OUTPUT, '*T')
  FOR I = 1 TO V*(0) DO WRITECH(OUTPUT, V*(I))
  $)
```

AND OUTSTRING(S) BE

```
$( LET V = VEC BYTEMAX
  UNPACKSTRING(H2*(S), V)
  OUT1(STRINGCONST)
  FOR I = 0 TO V*(0) DO OUT1(V*(I))
  UNLESS LISTING RETURN
  WRITECH(OUTPUT, '*T')
  FOR I = 1 TO V*(0) DO WRITECH(OUTPUT, V*(I))
  $)
```

AND OUTPSOP(L, OP, N) BE

```
$(1 OUT1(OP)
  OUT2(L, N)
  UNLESS LISTING RETURN
```

```
WRITES('*N*NL')
WRITEN(L)
WRITECH(OUTPUT, '*T')
WRITES(OP EQ EQU -* 'EQU', 'ERROR')
WRITECH(OUTPUT, '*T')
WRITEN(N) $)1
```

AND OUT1(X) BE

```
$( RV CODEFILEP := X
  CODEFILEP := CODEFILEP + 1
  IF CODEFILEP GE AETREEP DO OVERFLOW() $)
```

AND OUT2(X, Y) BE

```
$( CODEFILEP*(0) := X
  CODEFILEP*(1) := Y
  CODEFILEP := CODEFILEP + 2
  IF CODEFILEP GE AETREEP DO OVERFLOW() $)
```

AND OVERFLOW() BE

```
$( WRITES('*N*T***** POCODE STORAGE AREA OVERFLOW. ')
  WRITES('COMPILATION TERMINATED.*N*N')
  COMPERROR := TRUE
  LONGJUMP(ECP, ECPLEVEL) $)
```