```
   let Command[] be
   §C
       Read[]
    5  §r
           let x = Ch
           and OldNRD = NonRecDef
           and HN1, F, C = 0, 0, 0
           and RBHN, RLHN = BHN, LHN
   10      switchon ChType into
           §s
               case IF:
               case UNLESS:
                       HN1 := CondExpression[Ch]
   15                  unless Ch = DO do CompilerError[3301]
                       Command[]
                       ForHopPt[Abs[HN1]]
                       if HopNoUsed[HN1] do UpdateS[SSP]
                       break
   20
               case WHILE:
               case UNTIL:
                       HN1 := HopNo[]
                       BackHopPt[HN1]
   25                  BHN := CondExpression[Ch]
                       unless Ch = DO do CompilerError[3302]
                       LHN := UnusedHopNo[]
                       Command[]
                       ForHopPt[LHN]
   30                  if HopNoUsed[LHN] do UpdateS[SSP]
                       BackHop[HN1]
                       ForHopPt[Abs[BHN]]
                       if HopNoUsed[BHN] do UpdateS[SSP]
                       BHN, LHN := RBHN, RLHN
   35                  break

               case REPSTART:
                     § let HN2 = -UNSET
                       HN1 := HopNo[]
   40                  BHN, LHN := UnusedHopNo[], UnusedHopNo[]
                       BackHopPt[HN1]
                       Command[]
                       ForHopPt[LHN]
                       if HopNoUsed[LHN] do UpdateS[SSP]
   45                  switchon Ch into
                       §s
                           case REPEAT:
                                   Read[]
                                   BackHop[HN1]
   50                              endcase

                           case REPEATWHILE:
                           case REPEATUNTIL:
                                   HN2 := CondExpression[Ch]
   55                              BackHop[HN1]
                                   ForHopPt[Abs[HN2]]
                                   endcase

                           default:CompilerError[3303]
   60                  $s
                       ForHopPt[BHN]
```

```
                if HopNoUsed[HN2] ∨ HopNoUsed[BHN] do UpdateS[SSP]
                BHN, LHN := RBHN, RLHN
                break
65            §

          case TEST:
            § let HN2 = HopNo[]
              HN1 := CondExpression[Ch]
70            unless Ch = DO ∨ Ch = IFSO ∨ Ch = IFNOT do
                  CompilerError[3304]
              Command[]
              unless Ch = OR ∨ Ch = IFSO ∨ Ch = IFNOT do
                  CompilerError[3305]
75            ForHop[HN2]
              ForHopPt[Abs[HN1]]
              if HopNoUsed[HN1] do UpdateS[SSP]
              Command[]
              ForHopPt[HN2]
80            break
            §

          case FOR:
              HN1 := HopNo[]
85            Read[]
              Read[]
              F := FindName[Ch]
              StartNonRecDef[]
              Expression[]
90            unless Ch = TO do CompilerError[3306]
              UpdateNameVal[F, SSP + 1]
              Expression[]
              Inst[SWAPsm]
              BackHopPt[HN1]
95            Inst[DDUPsm]
              C := (Ch = BY → ConstantExpression[], 1)
              EndNonRecDef[OldNRD]
              unless Ch = DO do CompilerError[3307]
              BHN, LHN := HopNo[], UnusedHopNo[]
100           CondForHop[C ≥ 0 → HOPIFGTsm, HOPIFLTsm, BHN]
              Command[]
              ForHopPt[LHN]
              if HopNoUsed[LHN] do UpdateS[SSP]
              Code[INCsm, C]
105           BackHop[HN1]
              ForHopPt[BHN]
              BHN, LHN := RBHN, RLHN
              UpdateS[SSP - 2]
              unless Ch = ENDFOR do CompilerError[3308]
110           Read[]
              break

          case LASS:
              DealwithAss[]
115           break

          case SWITCHON:
            § let RSVec = SwitchVec
              and RSSP = SSP
120           and RCHN, REHN = CHN, EHN
              and RCaseGen = CaseGen
              CaseGen := Generation[PresentLevel]
              Read[]
              SetupSwitchVec[]
125           Expression[]
              unless Ch = INTO do CompilerError[3309]
```

```
                    Code[SWITCHsm, SwitchVec↓3]
                    Command[]
                    CellOutput[SwitchVec + 1]
130                 ForHopPt[EHN]
                    ReturnVec[SwitchVec, SwitchVec↓2]
                    SwitchVec := RSVec
                    UpdateS[RSSP]
                    CHN, EHN := RCHN, REHN
135                 CaseGen := RCaseGen
                    unless Ch = ENDSWITCH do CompilerError[3310]
                    Read[]
                    break
                  §
140
                case CASE:
                    C := ConstantExpression[]
                    test CHN=UNSET ∨ CaseGen≠Generation[PresentLevel]
                      ifso  Error[251, HARD, DUMMY, NullProgram]
145                   ifnot §   UpdateSwitchVec[C]
                                CHN := CHN + 1
                                ForHopPt[CHN]
                            §
                    Read[]
150                 loop

                case DEFAULT:
                    Read[]
                    test EHN=UNSET ∨ CaseGen≠Generation[PresentLevel]
155                   ifso  Error[255, HARD, DUMMY, NullProgram]
                      ifnot §   unless SwitchVec↓5 = EHN do
                                    Error[258, HARD, PresentLevel,
                                        NullProgram]
                                SwitchVec↓5 := EHN - 1
160                             ForHopPt[EHN - 1]
                            §
                    Read[]
                    loop

165             case ENDCASE:
                    if EHN = UNSET do
                        Error[252, HARD, DUMMY, NullProgram]
                    ForHop[EHN]
                    Read[]
170                 break

                case GLOBAL:
                case STATIC:
                case MANIFEST:
175             case TABLE:
                    Read[]
                  § let l, g = LineNo, GetNo
                    F := FindName[Ch]
                    StartNonRecDef[]
180                 C := ConstantExpression[]
                    switchon x into
                    §s
                        case STATIC:
                                test NameType[F] = GLOBAL
185                               ifso  §   SetLitGlobal[NameNo[F],
                                                NameVal[F], C]
                                            RememberGlobal[F]
                                        §
                                  ifnot §   SetLitStatic[NameNo[F], C]
190                                         CheckScope[F, l, g]
                                        §
```

```
                        endcase

                    case GLOBAL:
195                     unless 0 ≤ C ≤ 500 ∨
                                NameVal[F] = UNUSED do
                            Error[261, HARD, C, NullProgram]
                        UpdateNameVal[F, C]
                        CheckScope[F, l, g]
200                     endcase

                    case MANIFEST:
                        UpdateNameVal[F, C]
                        UpdateNameNo[F, SET]
205                     CheckScope[F, l, g]
                        endcase

                    case TABLE:
                          § let n = 1
210                       Write[C]
                          while Ch = COMMA do
                          §   C := ConstantExpression[]
                              n := n + 1
                              Write[C]
215                       §
                          TableMax := TableMax + n
                          test NameType[F] = GLOBAL
                            ifso   §   SetDSegGlobal[NameNo[F],
                                           NameVal[F], n]
220                                    RememberGlobal[F]
                                   §
                            ifnot  §   SetDSegStatic[NameNo[F], n]
                                       CheckScope[F, l, g]
                                   §
225                       endcase
                        §
                $s
                EndNonRecDef[OldNRD]
                loop
230         §

            case LOCDEF:
                  § let RSSP = SSP
                    Read[]
235                 §r
                        Read[]
                        F := FindName[Ch]
                        Read[]
                        SSP := SSP + 1
240                     UpdateNameVal[F, SSP]
                        unless Ch = COMMA break
                        Read[]
                    $r repeat
                    unless Ch = DEF do CompilerError[3311]
245                 SSP := RSSP
                    StartNonRecDef[]
                    Expression[] repeatuntil Ch ≠ COMMA
                    EndNonRecDef[OldNRD]
                    loop
250             §

            case VECDF:
                    Read[]
                    F := FindName[Ch]
255                 SSP := SSP + 1
                    UpdateNameVal[F, SSP]
```

```
                StartNonRecDef[]
                C := ConstantExpression[]
                if C < 0 do Error[262, HARD, DUMMY, NullProgram]
                UpdateNameNo[F, C]
                Code[LPsm, SSP + 1]
                SSP := SSP + C
                EndNonRecDef[OldNRD]
                UpdateS[SSP]
                loop

            case PARAM:
                §r
                    Read[]
                    F := FindName[Ch]
                    SSP := SSP + 1
                    UpdateNameVal[F, SSP]
                    Read[]
                    unless Ch = COMMA break
                    Read[]
                $r repeat
                if PresentOutput = MFDEF do ManFunDefParams[]
                loop

            case FNDF:
            case RTDF:
                  § let RSSP = SSP
                    and G = LocalGenNo
                    ClearStack[]
                    HN1 := HopNo[]
                    Read[]
                    F := FindName[Ch]
                    StartOutputMode[NORMAL]
                    ForHop[HN1]
                    SetCSegGloborStat[F]
                    SSP := -1
                    LocalGenNo := Generation[PresentLevel]
                    Command[]
                    unless Ch = ENDBODY do CompilerError[3312]
                    LocalGenNo := G
                    SSP := RSSP
                    ForHopPt[HN1]
                    EndOutputMode[]
                    Read[]
                    break
                  §

            case MANFNDF:
            case MANRTDF:
                    Read[]
                    ManFunDefinition[x]
                    Read[]
                    break

            case FNBODY:
            case RTBODY:
                  § let RVHN, RCHN, REHN = VHN, CHN, EHN
                    let RInRt = InRt
                    InRt := (Ch = RTBODY)
                    BHN, LHN, VHN, CHN, EHN :=
                        UNSET, UNSET, UNSET, UNSET, UNSET
                    Code[JUMPPTsm, SSP]
                    test Ch = FNBODY
                      ifso  §   Expression[]
                                Code[FNEXITsm, SSP]
                            §
```

```
                    ifnot §   Command[]
                              Code[RTEXITsm, SSP]
                          §
                BHN, LHN, VHN, CHN, EHN :=
                    RBHN, RLHN, RVHN, RCHN, REHN
                InRt := RInRt
                break
              §

        case BREAK:
                if BHN = UNSET do
                    Error[253, HARD, DUMMY, NullProgram]
                SpecForHop[lv BHN]
                Read[]
                break

        case LOOP:
                if LHN = UNSET do
                    Error[257, HARD, DUMMY, NullProgram]
                SpecForHop[lv LHN]
                Read[]
                break

        case RESULTIS:
                if VHN = UNSET do
                    Error[254, HARD, DUMMY, NullProgram]
                Expression[]
                unless VSSP = SSP do
                §   Code[SPsm, VSSP]
                    Code[Ssm, VSSP]
                    SSP := SSP + 1
                §
                SpecForHop[lv VHN]
                SSP := SSP - 1
                break

        case GOTO:
                Expression[]
                Inst[GOTOsm]
                break

        case RETURN:
                test InRt
                  ifnot Error[259, HARD, DUMMY, NullProgram]
                  ifso  Code[RETURNsm, SSP]
                Read[]
                break

        case VALOF:
                Valof[]
                RoutineApp[]
                break

        case RBRA:
                Addto[ShuntVec, Ch]
                Expression[]
                RoutineApp[]
                break

        case NAME:
        case NUMBER:
        case CHARACTER:
        case STRING:
        case TRUE:
        case FALSE:
```

```
                    RightVersion[Ch, ChType]
                    RoutineApp[]
                    break
390
            case LABEL:
                    Read[]
                    F := FindName[Ch]
                    SetCSegGloborStat[F]
395                 Code[JUMPPTsm, SSP]
                    Read[]
                    loop

            case SECTBRA:
400                 § let RSSP = SSP
                    Command[] repeatwhile Ch = SEMICOLON
                    unless ChType = SECTKET ∧
                            ValPart[Ch] = ValPart[x] do
                        CompilerError[3313]
405                 unless SSP = RSSP do UpdateS[RSSP]
                    Read[]
                    break
                  §

410         case LET:
            case AND:
                    Read[]
                    loop

415         default:
                    break
        $s
    $r repeat

420 switchon ChType into
    §s
        case LET:
        case AND:
        case SECTBRA:
425             BackUp[SEMICOLON]

        case SECTKET:
        case SEMICOLON:
        case OR:
430     case IFSO:
        case IFNOT:
        case REPEAT:
        case REPEATUNTIL:
        case REPEATWHILE:
435     case ENDBODY:
        case ENDFOR:
        case ENDVALOF:
        case ENDSWITCH:
        case ENDPROG:
440             return

        default:Error[105, HARD, Ch, DeleteCommand]
    $s
$C
445

and SpecForHop[Loc] be
§SFH
    if rv Loc ≤ 0 do rv Loc := - rv Loc
450 ForHop[rv Loc]
$SFH
```

```
    and SetCSegGloborStat[F] be
455     test NameType[F] = GLOBAL
          ifso  §   SetCSegGlobal[NameNo[F], NameVal[F], NameField[F]]
                    RememberGlobal[F]
                §
          ifnot §   UpdateNameVal[F, UNSET]
460               SetCSegStatic[NameNo[F], NameField[F]]
                  CheckScope[F, LineNo, GetNo]
                §


465 and SetupSwitchVec[] be
    §SSV
        let n = ValPart[Ch]
        let m = MaxVecSize[]
        unless n + 5 ≤ m do
470     §   Peep[2]
            Error[9, HARD, DUMMY, Finish]
        §
        SwitchVec := NewVec[n + 5]
        SwitchVec↓0 := 0
475     SwitchVec↓1 := SWITCH
        SwitchVec↓2 := n + 5
        SwitchVec↓3 := SwitchNo[]
        CHN := HN
        HN := HN + n + 1
480     EHN := HopNo[]
        SwitchVec↓4 := CHN + 1
        SwitchVec↓5 := EHN
    $SSV

485
    and UpdateSwitchVec[a] be
    §USV
        let p = SwitchVec↓0 + 6
        if p > SwitchVec↓2 do CompilerError[3314]
490     SwitchVec↓0 := p - 5
        SwitchVec↓p := a
        for i = 6 to p - 1 do
            if SwitchVec↓i = SwitchVec↓p do
            §   Error[256, HARD, PresentLevel, NullProgram]
495             break
            §
    $USV


500 and RoutineApp[] be
    §RA
        Read[]
        unless Ch = SBRA do
        §   Error[118, HARD, Ch, BackUp, ERROR]
505         return
        §
        Application[RTEXIT] repeatwhile Ch = SBRA
    $RA

510
    and RememberGlobal[F] be
    §RG
        let n = GlobList↓0 + 1
        and v = NameVal[F]
515     and i = ValPart[NameField[F]]
        let Sys = (0 ≤ v ≤ 99 ∧ v ≠ 1) ∨ (300 ≤ v ≤ 399)
```

```
        test n > GlobSize
          ifso  §   if n = GlobSize + 1 do
                        Error[20, SOFT, DUMMY, NullProgram]
520                 if Sys do Error[21, SOFT, i, NullProgram]
                  §
          ifnot GlobList↓(2 × n - 1), GlobList↓(2 × n) :=
                      v ∨ (Sys → 8100000, 0), i
        GlobList↓0 := n
525     if Sys do GlobList↓(-1) := GlobList↓(-1) + 1
    $RG

****
```