```
        LOCAL descr;                                          1110g1a
      %construct descriptor%                                  1110g2
        descr          = &src;                              1110g2a
        descr.ledtype  = lblock;                            1110g2b
        descr.ledalo   = TRUE;                              1110g2c
      %return%                                                1110g3
        RETURN (descr);                                     1110g3a
        END,                                                1110g3b

    (descrl) %construct list list descriptor%
    PROCEDURE (src REF %-> descr%);                           1110h
      %declarations%                                          1110h1
        LOCAL descr;                                        1110h1a
      %construct descriptor%                                  1110h2
        descr          = &src;                              1110h2a
        descr.ledtype  = llist;                             1110h2b
        descr.ledalo   = TRUE;                              1110h2c
      %return%                                                1110h3
        RETURN (descr);                                     1110h3a
        END,                                                1110h3b

%error management%                                            1111

    (sigerr) %DPS error%
    PROCEDURE (number);                                       1111a
      %declarations%                                          1111a1
        LOCAL msg;                                          1111a1a
      %advance meter%                                         1111a2
```

```
        BUMP vmderc;                                                    1111a2a
    %verify error number%                                               1111a3
        IF NOT (number IN [1, emxern]) THEN number _ eeuern;            1111a3a
        msg _ fnderm (number);                                          1111a3b
    %trace error%                                                       1111a4
        IF dtrace ,A ctrerrs THEN trcerr (number, msg);                 1111a4a
    %abort%                                                             1111a5
        ABORT (number, msg);                                            1111a5a
        END,
                                                                        1111a5b
(always) %compute UNLCKREC disposition%
PROCEDURE (disp %-> disp%);                                             1111b
    %return%                                                            1111b1
        RETURN (IF rtn () OR abr () THEN disp ELSE FALSE);              1111b1a
        END,
                                                                        1111b1b
(ifrtn) %compute UNLCKREC disposition%
PROCEDURE (disp %-> disp%);                                             1111c
    %return%                                                            1111c1
        RETURN                                                          1111c1a
            (IF rtn () THEN disp                                        1111c1a1
            ELSE                                                        1111c1a2
                IF abr () THEN disp ,X cdspmsk                          1111c1a2a
                ELSE FALSE);                                            1111c1a2b
        END,
                                                                        1111c1b
(abrtn) %abort or return?%
PROCEDURE %(-> outcome)%;                                               1111d
```

```
        %return%                                                    1111d1

           RETURN (rtn () OR abr ());                               1111d1a

           END,
                                                                    1111d1b

        (abr) %abort?%
        PROCEDURE %(=> outcome)%;                                   1111e

           %return%                                                 1111e1

           RETURN (SIGNALTYPE = aborttype);                         1111e1a

           END,
                                                                    1111e1b

        (rtn) %return?%
        PROCEDURE %(=> outcome)%;                                   1111f

           %return%                                                 1111f1

           RETURN (SIGNALTYPE = notetype AND SIGNAL = return);      1111f1a

           END,
                                                                    1111f1b

     %subroutines%                                                  1112

        %processes%                                                 1112a

           (chndir) %determine logical channel direction%
           PROCEDURE (sg POINTER, inph %=> outph, outsh, outdead%); 1112a1

              %declarations%                                        1112a1a

                 LOCAL outph, outsh, outdead;                       1112a1a1

              %determine channel direction%                         1112a1b

                 CASE inph OF                                       1112a1b1

                    = sg.sgph1:                                     1112a1b1a

                       BEGIN                                        1112a1b1a1

                       outph   _ sg.sgph2;                          1112a1b1a2

                       outsh   _ sg.sgsh2;                          1112a1b1a3
```

```
                outdead _ sg.sg2dead;                                   1i12a1b1a4
                END;                                                    1i12a1b1a5
            = sg.sgph2;                                                 1i12a1b1b
                BEGIN                                                   1i12a1b1b1
                outph   _ sg.sgph1;                                     1i12a1b1b2
                outsh   _ sg.sgsh1;                                     1i12a1b1b3
                outdead _ sg.sg1dead;                                   1i12a1b1b4
                END                                                     1i12a1b1b5
            ENDCASE sigerr (egmhsg);                                    1i12a1b1c
        %return%                                                        1i12a1c
            RETURN (outph, outsh, outdead);                             1i12a1c1
            END.
                                                                        1i12a1c2
%packages%                                                              1i12b

    (iopnpk) %open package%
    PROCEDURE (pkname REF, stupinfo REF, scope %-> pkh%);               1i12b1
        %declarations%                                                  1i12b1a
            LOCAL pkh, port, sph, i, use, hashcode, ub, inildr;
                                                                        1i12b1a1
            LOCAL pknames REF, pkhashs REF, pkuses REF;                 1i12b1a2
            LOCAL pk POINTER, su POINTER;                               1i12b1a3
        %create package record%                                         1i12b1b
            pk _ crtrec (vpkfld, scope : pkh);                          1i12b1b1
            INVOKE (prtnkp,, pk);                                       1i12b1b2
        %locate subprocess%                                             1i12b1c
            hashcode _ hash (&pkname);                                  1i12b1c1
            OPENPORT runfld (vsufld, copncls : [port]);                 1i12b1c2
```

```
      LOOP                                                    1112b1c3
         BEGIN                                                1112b1c3a
         WHILE su _ PCALL [port] (cprevue : sph) DO           1112b1c3b
            IF &pknames _ su.supknames THEN                   1112b1c3b1
               BEGIN                                          1112b1c3b1a
               &pkhashs _ su.supkhashs;                       1112b1c3b1b
               ub _ pknames.L;                                1112b1c3b1c
               FOR i _ 1 UP UNTIL > ub DO                     1112b1c3b1d
                  IF ELEM #pkhashs# [i] = hashcode            1112b1c3b1d1
                  AND *[ELEM #pknames# [i]]* = *pkname*
                                                              1112b1c3b1d2
                  AND PCALL [port] (clock) THEN               1112b1c3b1d3
                     EXIT LOOP 3;                             1112b1c3b1d3a
               END;                                           1112b1c3b1e
         sigerr (ekupkn);                                     1112b1c3c
         END;                                                 1112b1c3d
      %OK opening of package%                                 1112b1d
         use _ ELEM #[&pkuses _ su.supkuses]# [i];            1112b1d1
         inildr _ sph = vpsldr AND NOT use;                   1112b1d2
         vjusr (                                              1112b1d3
            vpsldr, 0, cokopk, pkh, &pkname, scope,
            &stupinfo, IF inildr THEN i ELSE 0);              1112b1d3a
      %initialize package if necessary%                       1112b1e
         #pkuses# [i] _ use + 1;                              1112b1e1
         IF NOT (use OR inildr) THEN                          1112b1e2
            vjusr (sph, 0, cinipk, i);                        1112b1e2a
```

```
        pk,pkintpkh _ i;                                        1112b1e3
        pk,pksph    _ sph;                                      1112b1e4
    %return%                                                    1112b1f
        RETURN (pkh);                                           1112b1f1
        END,                                                    1112b1f2

(iclspk) %close package%
PROCEDURE (pkh %-> cost%);                                      1112b2
    %declarations%                                              1112b2a
        LOCAL use, sph, intpkh, trmldr;                         1112b2a1
        LOCAL pkuses REF;                                       1112b2a2
        LOCAL pk POINTER, su POINTER;                           1112b2a3
    %locate package record%                                     1112b2b
        pk _ fndrec (vpkfld, pkh, cdelete);                     1112b2b1
        INVOKE (prtndl,, pk);                                   1112b2b2
    %locate subprocess record%                                  1112b2c
        su _ fndrec (vsufld, sph _ pk.pksph, copncls);          1112b2c1
        INVOKE (palwkp,, su);                                   1112b2c2
    %OK closing of package%                                     1112b2d
        use _                                                   1112b2d1
            ELEM #[&pkuses _ su,supkuses]# [intpkh _
            pk.pkintpkh] - 1;                                   1112b2d1a
        trmldr _ sph = vpsldr AND NOT use;                      1112b2d2
        vjusr (                                                 1112b2d3
            vpsldr, 0, cokcpk, ELEM #[su,supknames]#
            [intpkh], pkh, IF trmldr THEN intpkh ELSE 0);       1112b2d3a
    %flush package from system%                                 1112b2e
```

```
            pkflsh (pkh);                                          1i12b2e1

         %terminate package if necessary%                          1i12b2f

            #pkuses# [intpkh] _ use;                               1i12b2f1

            IF NOT (use OR trmidr) THEN                            1i12b2f2

               vjusr (sph, 0, ctrmpk, intpkh);                     1i12b2f2a

         %return%                                                  1i12b2g

            RETURN (pk,pkcost);                                    1i12b2g1

            END,                                                   1i12b2g2

      %procedures%                                                 1i12c

         (l10cal) %call L10 procedure%
         PROCEDURE (entry POINTER, arg1 REF, res1 REF);            1i12c1

            %declarations%                                         1i12c1a

               LOCAL i, ub, bp, spec, type;                        1i12c1a1

               LOCAL rawarg1 [cmarg1], rawres1 [cmres1];           1i12c1a2

            %decompose argument list%                              1i12c1b

               mkarray (&arg1, $rawarg1, FALSE);                   1i12c1b1

            %call procedure%                                       1i12c1c

               rawres1 _ [entry,enpadr]                            1i12c1c1

                  (rawarg1, rawarg1 [1], rawarg1 [2], rawarg1 [3],
                  rawarg1 [4], rawarg1 [5], rawarg1 [6], rawarg1
                  [7] : rawres1 [1], rawres1 [2]);                 1i12c1c1a

            %construc result list%                                 1i12c1d

               IF &res1 THEN                                       1i12c1d1

                  BEGIN                                            1i12c1d1a

                  ub _ entry,enresc;                               1i12c1d1b

                  bp _ opntwrd (entry+4, cprpmspc1);               1i12c1d1c
```

```
            FOR i _ 1 UP UNTIL > ub DO                          1112c1d1d
               BEGIN                                            1112c1d1d1
               spec _ ordbyt ($bp);                             1112c1d1d2
               type _ spec.spdtype;                             1112c1d1d3
               IF type = calostruc OR type = calodata THEN
                                                                1112c1d1d4
                  #resl# !_ USE descrb (rawresl [i-1])          1112c1d1d4a
               ELSE #resl# !_ rawresl [i-1];                    1112c1d1d5
               END;                                             1112c1d1d6
            END;                                                1112c1d1e
         %return%                                               1112c1e
            RETURN;                                             1112c1e1
            END.
                                                                1112c1e2
      %+PDP10% (mskarlst) %mask argument/result list%
      PROCEDURE (type, prml REF, mask REF %-> prml REF%);       1112c2
         %declarations%                                         1112c2a
            LOCAL count, i, maski, ub;                          1112c2a1
            LOCAL LIST dsel [cmdsell];                          1112c2a2
         %catchphrases%                                         1112c2b
            (pnulst) CATCHPHRASE; IF abrtn () THEN NULL-LISTS;
                                                                1112c2b1
         %assure local lists released%                          1112c2c
            INVOKE (pnulst);                                    1112c2c1
         %NOP if no mask%                                       1112c2d
            IF NOT &mask THEN RETURN (&prml);                   1112c2d1
         %loop through list%                                    1112c2e
            ub _ prml.L;                                        1112c2e1
```

```
count _ mask.L;                                          1112c2e2

FOR i _ 1 UP UNTIL > ub DO                               1112c2e3

   IF [maski _ ELEM #mask# [MIN (i, count)]].dstype
   = cindex THEN                                         1112c2e3a

      CASE decstruc (cindex, maski, 0) OF                1112c2e3a1

         = ccaller: NULL;                                1112c2e3a1a

         = cdiscard:                                     1112c2e3a1b

            IF type = creslist THEN                      1112c2e3a1b1

               #prml# [i] _                              1112c2e3a1b1a

                  USE descrb (encstruc (cempty))
                                                         1112c2e3a1b1a1
            ELSE REPEAT CASE (0);                        1112c2e3a1b2

         ENDCASE sigerr (epimsk)                         1112c2e3a1c

   ELSE                                                  1112c2e3b

      BEGIN                                              1112c2e3b1

      decstruc (cdsel, maski, $dsel);                    1112c2e3b2

      IF type = carglist THEN                            1112c2e3b3

         #prml# [i] _ USE descrb (rddt ($dsel))
                                                         1112c2e3b3a
      ELSE                                               1112c2e3b4

         BEGIN                                           1112c2e3b4a

         wrdt ($dsel, ELEM #prml# [i]);                  1112c2e3b4b

         #prml# [i] _ USE descrb (encstruc
         (cempty));                                      1112c2e3b4c

         END;                                            1112c2e3b4d

      END;                                               1112c2e3b5

%return%                                                 1112c2f

   RETURN;                                               1112c2f1
```

```
            END,%+PDP10%                                                    1112c2f2

    %data stores%                                                           1112d

        %+PDP10% (icrtdt) %create data store record%
        PROCEDURE (dsel REF, scope %-> dt POINTER%);                        1112d1

            %declarations%                                                  1112d1a

                LOCAL dsname;                                               1112d1a1

                LOCAL dt POINTER;                                           1112d1a2

            %create data store record%                                      1112d1b

                dt _ crtrec (vdtfld, scope);                                1112d1b1

                INVOKE (puldldt,, dt);                                      1112d1b2

                dt,dtpkh  _ ELEM #dsel# [2];                                1112d1b3

                dt,dtname _                                                 1112d1b4

                    savent (ccharstr, dsname _ ELEM #dsel# [3]);            1112d1b4a

                dt,dthash _ hash (dsname);                                  1112d1b5

            %return%                                                        1112d1c

                dt,dtuse _ 1;                                               1112d1c1

                RETURN (dt);                                                1112d1c2

                END,%+PDP10%                                                1112d1c3

    %locks%                                                                 1112e

        (winscope) %verify one processor within scope of another%
        PROCEDURE (pcrid1 REF, scope, pcrid2 REF %-> outcome%);             1112e1

            %local%                                                         1112e1a

                LOCAL outcome;                                              1112e1a1

            %compare processor ids%                                         1112e1b

                outcome _                                                   1112e1b1
```

```
            CASE scope OF                                          1i12e1b1a
               = call: TRUE;                                       1i12e1b1a1
               = cprocess:                                         1i12e1b1a2
                  pcrid1 = pcrid2;                                 1i12e1b1a2a
               = csubprocess:                                      1i12e1b1a3
                  pcrid1 = pcrid2 AND pcrid1 [1] = pcrid2
                  [1];                                             1i12e1b1a3a
               = cprocessor:                                       1i12e1b1a4
                  pcrid1 = pcrid2 AND pcrid1 [1] = pcrid2
                  [1]
                  AND pcrid1 [2] = pcrid2 [2];                     1i12e1b1a4a
               ENDCASE sigerr (eruscp);                            1i12e1b1a5
         %return%                                                  1i12e1c
            RETURN (vwheel OR outcome);                            1i12e1c1
            END,
                                                                   1i12e1c2
      (okwone) %check two locks for compatibility%
      PROCEDURE (lckida REF, pcrida REF, lckidb REF, pcridb REF
      %-> outcome%);                                               1i12e2
         %declarations%                                            1i12e2a
            LOCAL same;                                            1i12e2a1
         %return%                                                  1i12e2b
            RETURN (                                               1i12e2b1
               %different locks%                                   1i12e2b1a
                  (lckida # lckidb) OR                             1i12e2b1a1
               %both share%                                        1i12e2b1b
                  ((same _ lckida [1] = lckidb [1])                1i12e2b1b1
                     AND lckida [1] = cshare) OR                   1i12e2b1b1a
```

```
            %different and neither exclusive%                    1i12e2b1c
               (NOT same                                         1i12e2b1c1
                  AND lckida [1] # cexclusive                    1i12e2b1c1a
                  AND lckidb [1] # cexclusive) OR                1i12e2b1c1b
               %within one another's scope%                      1i12e2b1d
                  winscope (&pcrida, MIN (lckida [2], lckidb
                  [2]), &pcridb));                               1i12e2b1d1
            END,
                                                                 1i12e2b2
         (okwall) %check lock request for compatibility with
         existing locks%
         PROCEDURE (lckid REF, pcrid REF %-> outcome%);          1i12e3
            %declarations%                                       1i12e3a
               LOCAL port, qpcrid, outcome=TRUE;                 1i12e3a1
               LOCAL qlckid [cmlkidl];                           1i12e3a2
               LOCAL ls POINTER, lcb POINTER;                    1i12e3a3
            %check against locks recorded in folder%             1i12e3b
               lcb _ lckid;                                      1i12e3b1
               IF lcb.lcfdsh OR lcb.lcfdex OR lcb.lcfdsp THEN    1i12e3b2
                  BEGIN                                          1i12e3b2a
                  OPENPORT runfld (vlsfld, FALSE : [port]);      1i12e3b2b
                  WHILE ls _ PCALL [port] (clock) DO             1i12e3b2c
                     IF ls.lslocked                              1i12e3b2c1
                     AND NOT                                     1i12e3b2c2
                        okwone (&lckid, &pcrid, ls.lslckid,
                        ls.lspcrid)                              1i12e3b2c2a
                     AND NOT outcome _ FALSE THEN EXIT LOOP;     1i12e3b2c3
```

```
        END;                                                    1i12e3b2d

    %check against locks recorded in LCB%                       1i12e3c

        qlckid      _ &lcb;                                     1i12e3c1

        qlckid [2] _ cprocessor;                                1i12e3c2

        qpcrid _ lcb.lcpcrid;                                   1i12e3c3

        outcome _ outcome                                       1i12e3c4

            AND (NOT (lcb.lcimex AND qlckid [1] _
            cexclusive)                                         1i12e3c4a

                OR okwone (&lckid, &pcrid, $qlckid, qpcrid))
                                                            1i12e3c4a1
            AND (NOT (lcb.lcimsh AND qlckid [1] _ cshare)   1i12e3c4b

                OR okwone (&lckid, &pcrid, $qlckid, qpcrid));
                                                            1i12e3c4b1
    %return%                                                    1i12e3d

        RETURN (outcome);                                       1i12e3d1

        END.
                                                                1i12e3d2

(nodedlck) %check lock request for freedom from deadlock%
PROCEDURE (lckid REF, pcrid REF, endlckid REF, endpcrid
REF);                                                           1i12e4

    %declarations%                                              1i12e4a

        LOCAL port1, port2, lckid1, lckid2, pcrid1, pcrid2;
                                                            1i12e4a1
        LOCAL ls1 POINTER, ls2 POINTER;                     1i12e4a2

    %temp%                                                      1i12e4b

        RETURN;                                             1i12e4b1

    %open ports%                                                1i12e4c

        OPENPORT runfld (vlsfld, FALSE : [port1]);          1i12e4c1

        OPENPORT runfld (vlsfld, FALSE : [port2]);          1i12e4c2

    %check for deadlock%                                        1i12e4d
```

```
            WHILE ls1 _ PCALL [port1] (clock) DO                    1112e4d1
               IF ls1.lslocked THEN                                 1112e4d1a
                  IF NOT okwone (lckid1 _ ls1.lslckid, pcrid1 _
                  ls1.lspcrid, &endlckid, &endpcrid) THEN           1112e4d1a1
                     sigerr (elfded)                                1112e4d1a1a
                  ELSE                                              1112e4d1a2
                     BEGIN                                          1112e4d1a2a
                     PCALL [port2] (crewind);                       1112e4d1a2b
                     WHILE ls2 _ PCALL [port2] (clock) DO           1112e4d1a2c
                        IF NOT ls2.lslocked                         1112e4d1a2c1
                        AND NOT                                     1112e4d1a2c2
                           okwone (lckid2 _ ls2.lslckid, pcrid2
                           _ ls2.lspcrid, lckid1, pcrid1)
                                                                    1112e4d1a2c2a
                        THEN                                        1112e4d1a2c3
                           nodedlck (lckid2, pcrid2, &endlckid,
                           &endpcrid);                              1112e4d1a2c3a
                     END;                                           1112e4d1a2d
         %return%                                                   1112e4e
            RETURN;                                                 1112e4e1
            END.
                                                                    1112e4e2
%timers%                                                            1112f
   %+PDP10% (updtmr) %update virtual interval timers%
   PROCEDURE;                                                       1112f1
      %declarations%                                                1112f1a
         LOCAL                                                      1112f1a1
            port, code, tmh, evh, gone, left,
            interval=377777777778;                                  1112f1a1a
```

```
      LOCAL tm POINTER;                                          1i12f1a2
    %cancel system timer%                                        1i12f1b
      gone _ otsttmr (TRUE : left);                              1i12f1b1
    %update virtual timers%                                      1i12f1c
      OPENPORT runfld (vtmfld, FALSE : [port]);                  1i12f1c1
      WHILE tm _ PCALL [port] (clock : tmh) DO                   1i12f1c2
        IF (left _ tm,tmleft - gone) <= 0 THEN                   1i12f1c2a
          BEGIN                                                  1i12f1c2a1
          IF evh _ tm,tmevh THEN                                 1i12f1c2a2
            BEGIN                                                1i12f1c2a2a
            code,LH _ tmh; code,RH _ tm,tminterval;
                                                                 1i12f1c2a2b
            sigev (evh, code, TRUE);                             1i12f1c2a2c
            END                                                  1i12f1c2a2d
          ELSE sigecb (tm,tmecb, cok);                           1i12f1c2a3
          delrec (tm);                                           1i12f1c2a4
          END                                                    1i12f1c2a5
        ELSE interval _ MIN (interval, tm,tmleft _
        left);                                                   1i12f1c2b
    %reset system timer%                                         1i12f1d
      IF interval # 377777777777B THEN osettmr
      (interval);                                                1i12f1d1
    %return%                                                     1i12f1e
      RETURN;                                                    1i12f1e1
      END,%+PDP10%
                                                                 1i12f1e2
  %data structure conversion%                                    1i12g
```

```
(encfstruc) %encode formal data structure%
PROCEDURE (type, src REF %-> dst REF%);                    1i12g1

   %declarations%                                          1i12g1a

      LOCAL len, intlen, i, nxtsrc, nxtdst;                1i12g1a1

      LOCAL dst REF =0;                                    1i12g1a2

   %dispatch on basis of data type%                        1i12g1b

      INVOKE (pabrrb,, &dst);                              1i12g1b1

      CASE type OF                                         1i12g1b2

         = cempty:                                         1i12g1b2a

            BEGIN                                          1i12g1b2a1

            &dst _ aloent (cblock, 1);                     1i12g1b2a2

            len _ 0;                                       1i12g1b2a3

            END;                                           1i12g1b2a4

         = cboolean:                                       1i12g1b2b

            BEGIN                                          1i12g1b2b1

            &dst _ aloent (cblock, 1);                     1i12g1b2b2

            len _ IF &src THEN 1 ELSE 0;                   1i12g1b2b3

            END;                                           1i12g1b2b4

         = cindex:                                         1i12g1b2c

            BEGIN                                          1i12g1b2c1

            IF NOT (&src IN [1, cmxidx]) THEN              1i12g1b2c2

               sigerr (ediidx);                            1i12g1b2c2a

            &dst _ aloent (cblock, 1);                     1i12g1b2c3

            len _ &src;                                    1i12g1b2c4

            END;                                           1i12g1b2c5
```

```
= cinteger:                                          1i12g1b2d
   BEGIN                                             1i12g1b2d1
   &dst _ aloent (cblock, 2);                        1i12g1b2d2
   dst [1] _ &src;                                   1i12g1b2d3
   len _ 0;                                          1i12g1b2d4
   END;                                              1i12g1b2d5
= cbitstr:                                           1i12g1b2e
   BEGIN                                             1i12g1b2e1
   &dst _ savent (cblock, &src);                     1i12g1b2e2
   len _ src.dslen;                                  1i12g1b2e3
   END;                                              1i12g1b2e4
= ccharstr:                                          1i12g1b2f
   BEGIN                                             1i12g1b2f1
   &dst _                                            1i12g1b2f2
      aloent (cblock, (1 + (intlen _ ((len _
      src.L)+4)/5)));                                1i12g1b2f2a
   oblkxfr (&src+1, &dst+1, intlen);                 1i12g1b2f3
   END;                                              1i12g1b2f4
= clist:                                             1i12g1b2g
   BEGIN                                             1i12g1b2g1
   %compute size of list%                            1i12g1b2g2
      intlen _ 1;                                    1i12g1b2g2a
      len _ src.L;                                   1i12g1b2g2b
      FOR i _ 1 UP UNTIL > len DO                    1i12g1b2g2c
         intlen _ intlen +                           1i12g1b2g2c1
```

```
                    sizent (cblock, ELEM #src# [i]);                1112g1b2g2c1a
                %allocate space for it%                             1112g1b2g3
                   &dst _ aloent (cblock, intlen);                  1112g1b2g3a
                %copy the elements into it%                         1112g1b2g4
                   nxtdst _ &dst + 1;                               1112g1b2g4a
                   FOR i _ 1 UP UNTIL > len DO                      1112g1b2g4b
                      BEGIN                                         1112g1b2g4b1
                      nxtsrc _ ELEM #src# [i];                      1112g1b2g4b2
                      intlen _ sizent (cblock, nxtsrc);             1112g1b2g4b3
                      oblkxfr (nxtsrc,                              1112g1b2g4b4
                         nxtdst := nxtdst + intlen,                 1112g1b2g4b4a
                         intlen);                                   1112g1b2g4b4b
                      END;                                          1112g1b2g4b5
                END;                                                1112g1b2g5
             ENDCASE sigerr (edufty);                               1112g1b2h
          %complete data structure%                                 1112g1c
             dst.dstype _ type;                                     1112g1c1
             dst.dslen  _ len;                                      1112g1c2
          %return%                                                  1112g1d
             RETURN (&dst);                                         1112g1d1
             END,
                                                                    1112g1d2
(decfstruc) %decode formal data structure%
PROCEDURE (type, src REF, optdst REF %=> dst REF%);                 1112g2
   %declarations%                                                   1112g2a
      LOCAL                                                         1112g2a1
```

```
        len, i, nxtsrc, nxtdst, nxtlen, blklen,
        dstent=0, descr = (&optdst = -1);                    1i12g2a1a

    LOCAL dst REF = &optdst;                                 1i12g2a2

%catchphrases%                                               1i12g2b

    (prelent) CATCHPHRASE;                                   1i12g2b1

        IF abr () AND dstent AND &dst # &optdst THEN         1i12g2b1a

            relent (dstent, &dst);                           1i12g2b1a1

%verify data type%                                           1i12g2c

    IF type # src.dstype THEN sigerr (edwtyp);               1i12g2c1

%dispatch on basis of data type%                             1i12g2d

    len _ src.dslen;                                         1i12g2d1

    INVOKE (prelent);                                        1i12g2d2

    CASE type OF                                             1i12g2d3

        = cempty, = cboolean, = cindex:                      1i12g2d3a

            &dst _ IF descr                                  1i12g2d3a1

                THEN makelm (linteg, len)                    1i12g2d3a1a

                ELSE len;                                    1i12g2d3a1b

        = cinteger:                                          1i12g2d3b

            &dst _ IF descr                                  1i12g2d3b1

                THEN makelm (linteg, src [1])                1i12g2d3b1a

                ELSE src [1];                                1i12g2d3b1b

        = cbitstr:                                           1i12g2d3c

            BEGIN                                            1i12g2d3c1

            &dst _ savent (dstent _ cblock, &src);           1i12g2d3c2

            IF descr THEN &dst _ descrb (&dst);              1i12g2d3c3
```

```
    END;                                                    1112g2d3c4

= ccharstr:                                                 1112g2d3d

    BEGIN                                                   1112g2d3d1

    %allocate destination%                                  1112g2d3d2

       IF &optdst > 0 THEN                                  1112g2d3d2a

          IF len > optdst,M THEN sigerr (edostr)
                                                            1112g2d3d2a1
          ELSE NULL                                         1112g2d3d2a2

       ELSE &dst _ aloent (dstent _ ccharstr,
       len);                                                1112g2d3d2b

    %copy string to destination%                            1112g2d3d3

       oblkxfr (&src+1, &dst+1, (len+4)/5);                 1112g2d3d3a

       dst,L _ len;                                         1112g2d3d3b

    %construct descriptor%                                  1112g2d3d4

       IF descr THEN &dst _ descrb (&dst);                  1112g2d3d4a

    END;                                                    1112g2d3d5

= clist;                                                    1112g2d3e

    BEGIN                                                   1112g2d3e1

    IF &optdst > 0 THEN                                     1112g2d3e2

       BEGIN                                                1112g2d3e2a

       IF len > optdst,M THEN sigerr (edodcl);
                                                            1112g2d3e2b
       #optdst# _;                                          1112g2d3e2c

       END                                                  1112g2d3e2d

    ELSE &dst _ aloent (dstent _ clist, len);               1112g2d3e3

    nxtsrc _ &src + 1;                                      1112g2d3e4

    FOR i _ 1 UP UNTIL > len DO                             1112g2d3e5
```

```
                    BEGIN                                          1i12g2d3e5a
                    #dst# !_ USE descrb (nxtdst _                  1i12g2d3e5b
                       aloent (cblock, nxtlen _ sizstruc
                       (nxtsrc)));                                 1i12g2d3e5b1
                    oblkxfr (nxtsrc := nxtsrc + nxtlen,
                    nxtdst, nxtlen);                               1i12g2d3e5c
                    END;                                           1i12g2d3e5d
                 IF descr THEN &dst _ descrl (&dst);               1i12g2d3e6
                 END;                                              1i12g2d3e7
              ENDCASE sigerr (edufty);                             1i12g2d3f
           %return%                                                1i12g2e
              RETURN (&dst);                                       1i12g2e1
              END,
                                                                   1i12g2e2
        (encistruc) %encode informal data structure%
        PROCEDURE (type, src REF %-> dst REF%);                    1i12g3
           %declarations%                                          1i12g3a
              LOCAL dst, i;                                        1i12g3a1
              LOCAL entry POINTER;                                 1i12g3a2
           %dispatch via fundamental type%                         1i12g3b
              IF type = cstruc THEN                                1i12g3b1
                 dst _ savent (cblock, &src)                       1i12g3b1a
              ELSE                                                 1i12g3b2
                 BEGIN                                             1i12g3b2a
                 %locate table entry%                              1i12g3b2b
                    entry _ CASE type OF                           1i12g3b2b1
                       = cesel:    0;                              1i12g3b2b1a
```

```
                = cmessage: fndmsg (ELEM #src# [cmnumb]);          1i12g3b2b1b
                ENDCASE     fnddsd (type);                         1i12g3b2b1c

            %encode list elements%                                 1i12g3b2c

              IF entry THEN                                        1i12g3b2c1

                BEGIN                                              1i12g3b2c1a

                dst _ aloent (clist, src,L);                       1i12g3b2c1b

                INVOKE (palwrl,, dst);                             1i12g3b2c1c

                encprms (&src, entry,enargc, entry+2,
                dst);                                              1i12g3b2c1d

                END                                                1i12g3b2c1e

              ELSE dst _ &src;                                     1i12g3b2c2

            %encode list%                                          1i12g3b2d

              dst _ encstruc (clist, dst);                         1i12g3b2d1

            END;                                                   1i12g3b2e

        %return%                                                   1i12g3c

          RETURN (dst);                                            1i12g3c1

          END,
                                                                   1i12g3c2

(decistruc) %decode informal data structure%
PROCEDURE (type, src REF, optdst REF %-> dst REF%);                1i12g4

    %declarations%                                                 1i12g4a

      LOCAL i, descr = (&optdst = -1);                             1i12g4a1

      LOCAL dst REF, dstaddr REF =0;                               1i12g4a2

      LOCAL entry POINTER;                                         1i12g4a3

    %dispatch on basis of data type%                               1i12g4b

      CASE type OF                                                 1i12g4b1
```

```
= cstruc:                                                    1i12g4b1a
   BEGIN                                                     1i12g4b1a1
   &dst _ savent (cblock, &src);                             1i12g4b1a2
   IF descr THEN &dst _ descrb (&dst);                       1i12g4b1a3
   END;                                                      1i12g4b1a4
= cucstr:                                                    1i12g4b1b
   BEGIN                                                     1i12g4b1b1
   &dstaddr,RH _ &dst _                                      1i12g4b1b2
      decstruc (ccharstr, &src, &optdst);                    1i12g4b1b2a
   #dstaddr# _ + SF(#dstaddr#) SE(#dstaddr#);
                                                             1i12g4b1b3
   END;                                                      1i12g4b1b4
ENDCASE                                                      1i12g4b1c
   BEGIN                                                     1i12g4b1c1
   %decode list%                                             1i12g4b1c2
      &dstaddr,RH _ &dst _                                   1i12g4b1c2a
         decstruc (clist, &src, &optdst);                    1i12g4b1c2a1
      IF &dstaddr # &optdst THEN                             1i12g4b1c2b
         INVOKE (pabrrl,, &dstaddr);                         1i12g4b1c2b1
   %locate table entry%                                      1i12g4b1c3
      entry _ CASE type OF                                   1i12g4b1c3a
         = cesel: 0;                                         1i12g4b1c3a1
         = cmessage:                                         1i12g4b1c3a2
            fndmsg (decstruc (cindex, ELEM
            #dstaddr# [cmnumb], 0));                         1i12g4b1c3a2a
         ENDCASE fnddsd (type);                              1i12g4b1c3a3
```

```
                        %decode list elements%                          1i12g4b1c4

                           IF entry THEN                                1i12g4b1c4a

                              decprms (&dstaddr, entry,enargc,
                              entry+2);                                 1i12g4b1c4a1

                        END;                                            1i12g4b1c5

                  %return%                                              1i12g4c

                     RETURN (&dst);                                     1i12g4c1

                     END,
                                                                        1i12g4c2

%record appendages%                                                     1j

   %calls%                                                              1j1

      (cainit) %initialize call record%
      PROCEDURE (ca POINTER);                                           1j1a

         %declarations%                                                 1j1a1

            LOCAL ecb REF;                                              1j1a1a

         %initialize acquisition ECB%                                   1j1a2

            ca,caqecb _ &ecb _ ca + car,SIZE;                           1j1a2a

            ecb _ 1B6;                                                  1j1a2b

         %initialize acquisition ack ECB%                               1j1a3

            ca,carecb _ &ecb _ &ecb + 2;                                1j1a3a

            ecb _ 1B6;                                                  1j1a3b

         %return%                                                       1j1a4

            RETURN;                                                     1j1a4a

            END,
                                                                        1j1a4b

      (caterm) %terminate call record%
      PROCEDURE (ca POINTER);                                           1j1b
```

```
        %release attached storage%                                  1j1b1
           relent (cblock, ca,capname);                             1j1b1a
           relent (clist,  ca,caprml);                              1j1b1b
           relent (clist,  ca,camskl);                              1j1b1c
        %return%                                                    1j1b2
           RETURN;                                                  1j1b2a
           END,                                                     1j1b2b
     %no handle transformation%                                     1j1c
     %no flush%                                                     1j1d
  %+PDP10% %channels%                                               1j2
     %no initialization%                                            1j2a
     %no termination%                                               1j2b
     %no handle transformation%                                     1j2c
     (cnflsh) %flush channel from system%
     PROCEDURE (pch);                                               1j2d
        %declarations%                                              1j2d1
           LOCAL port;                                              1j2d1a
           LOCAL sg POINTER;                                        1j2d1b
        %verify channel unassociated with introduction%             1j2d2
           vwhlpls _ TRUE;                                          1j2d2a
           OPENPORT runfld (vsgfld, FALSE : [port]);                1j2d2b
           WHILE sg _ PCALL [port] (clock) DO                       1j2d2c
```

```
            IF sg.sgpch = pch THEN sigerr (egmhcn);                 1j2d2c1

        %return%                                                    1j2d3

            RETURN;                                                 1j2d3a

            END,%+PDP10%                                            1j2d3b

%+PDP10% %data stores%                                              1j3

    (dtinit) %initialize data store record%
    PROCEDURE (dt POINTER);                                         1j3a

        %initialize LCB%                                            1j3a1

            inilcb (dt,dtlcb _ dt + dtr.SIZE);                      1j3a1a

        %return%                                                    1j3a2

            RETURN;                                                 1j3a2a

            END,                                                    1j3a2b

    (dtterm) %terminate data store record%
    PROCEDURE (dt POINTER);                                         1j3b

        %release attached storage%                                  1j3b1

            relent (ccharstr, dt,dtname);                           1j3b1a

            relent (cblock,   dt,dtvalue);                          1j3b1b

        %terminate LCB%                                             1j3b2

            trmlcb (dt,dtlcb);                                      1j3b2a

        %return%                                                    1j3b3

            RETURN;                                                 1j3b3a

            END,                                                    1j3b3b

    %no handle transformation%                                      1j3c
```

```
    %no flush%%+PDP10%                                              1j3d

%+PDP10% %events%                                                   1j4

    %no initialization%                                             1j4a

    (evterm) %terminate event record%
    PROCEDURE (ev POINTER);                                         1j4b

      %release attached storage%                                    1j4b1

        relent (cblock, ev.evecb);                                  1j4b1a

      %return%                                                      1j4b2

        RETURN;                                                     1j4b2a

        END,                                                        1j4b2b

    %no handle transformation%                                      1j4c

    (evflsh) %flush event from system%
    PROCEDURE (evh);                                                1j4d

      %declarations%                                                1j4d1

        LOCAL port;                                                 1j4d1a

        LOCAL pr POINTER, ca POINTER, tm POINTER;                   1j4d1b

      %verify event unused for JUSR%                                1j4d2

        vwhlpls _ TRUE;                                             1j4d2a

        OPENPORT runfld (vprfld, FALSE : [port]);                   1j4d2b

        WHILE pr _ PCALL [port] (clock) DO                          1j4d2c

          IF pr.prjuevh = evh THEN sigerr (evfjuv);                 1j4d2c1

      %verify event unassociated with procedure call%               1j4d3

        OPENPORT runfld (vcafld, FALSE : [port]);                   1j4d3a

        WHILE ca _ PCALL [port] (clock) DO                          1j4d3b
```

```
            IF ca.caqevh = evh THEN sigerr (evfacq);                1j4d3b1
        %verify event unassociated with timer%                      1j4d4
            OPENPORT runfld (vtmfld, FALSE : [port]);               1j4d4a
            WHILE tm _ PCALL [port] (clock) DO                      1j4d4b
                IF tm.tmevh = evh THEN sigerr (evftmr);             1j4d4b1
        %return%                                                    1j4d5
            RETURN;                                                 1j4d5a
            END,%+PDP10%
                                                                    1j4d5b
    %locks%                                                         1j5
        (lkinit) %initialize lock record%
        PROCEDURE (lk POINTER);                                     1j5a
            %initialize LCB%                                        1j5a1
                inilcb (lk,lklcb _ lk + lkr,SIZE);                  1j5a1a
            %return%                                                1j5a2
                RETURN;                                             1j5a2a
                END,
                                                                    1j5a2b
        (lkterm) %terminate lock record%
        PROCEDURE (lk POINTER);                                     1j5b
            %terminate LCB%                                         1j5b1
                trmlcb (lk,lklcb);                                  1j5b1a
            %return%                                                1j5b2
                RETURN;                                             1j5b2a
                END,
                                                                    1j5b2b
        %no handle transformation%
                                                                    1j5c
```

```
    %no flush%                                                          1j5d

%lock sets%                                                             1j6

    (lsinit) %initialize lock set record%
    PROCEDURE (ls POINTER);                                             1j6a

      %declarations%                                                    1j6a1

        LOCAL lckid;                                                    1j6a1a

      %initialize lock id%                                              1j6a2

        ls.lslckid _ lckid _ ls + lsr.SIZE;                             1j6a2a

      %initialize processor id%                                         1j6a3

        ls.lspcrid _ lckid + cmlkid1;                                   1j6a3a

      %return%                                                          1j6a4

        RETURN;                                                         1j6a4a

        END,                                                            1j6a4b

    %no termination%                                                    1j6b

    %no handle transformation%                                          1j6c

    %no flush%                                                          1j6d

%managers%                                                              1j7

    (mginit) %initialize manager record%
    PROCEDURE (mg POINTER);                                             1j7a

      %allocate DPS state space%                                        1j7a1

        mg.mgstate _ aloent (cblock, $vstalen-$vstate);                 1j7a1a

      %initialize addr of ACs%                                          1j7a2

        mg.mgacs _ mg + mgr.SIZE;                                       1j7a2a
```

```
        %return%                                                    1j7a3

           RETURN;                                                  1j7a3a

           END,
                                                                    1j7a3b

        (mgterm) %terminate manager record%
        PROCEDURE (mg POINTER);                                     1j7b

           %release attached storage%                               1j7b1

              relent (cblock, mg,mgstate);                          1j7b1a

           %return%                                                 1j7b2

              RETURN;                                               1j7b2a

              END,
                                                                    1j7b2b

        %no handle transformation%
                                                                    1j7c

        %no flush%
                                                                    1j7d

     %packages%                                                     1j8

        %no initialization%
                                                                    1j8a

        %no termination%
                                                                    1j8b

        %no handle transformation%
                                                                    1j8c

        (pkflsh) %flush package from system%
        PROCEDURE (pkh);                                            1j8d

           %declarations%                                           1j8d1

              LOCAL port;                                           1j8d1a

              LOCAL LIST dsel [cmdsell];                            1j8d1b

              LOCAL dt POINTER;                                     1j8d1c
```

```
        %catchphrases%                                                  1j8d2
           (pnulst) CATCHPHRASE; IF abrtn () THEN NULL-LISTS;           1j8d2a
        %assure local lists released%                                   1j8d3
           INVOKE (pnulst);                                             1j8d3a
        %flush created data stores associated with package%             1j8d4
           vwhlpls _ TRUE;                                              1j8d4a
           OPENPORT runfld (vdtfld, cdelete : [port]);                  1j8d4b
           WHILE dt _ PCALL [port] (cprevue) DO                         1j8d4c
              IF dt.dtpkh = pkh AND PCALL [port] (clock) THEN           1j8d4c1
                 BEGIN                                                  1j8d4c1a
                 #dsel# _                                               1j8d4c1b
                    vfph, pkh, USE descrb (savent (ccharstr,
                    dt,dtname)), 0;                                     1j8d4c1b1
                 xdeldt ($dsel);                                        1j8d4c1c
                 END;                                                   1j8d4c1d
        %return%                                                        1j8d5
           RETURN;                                                      1j8d5a
           END,
                                                                        1j8d5b
%ports%                                                                 1j9
   (poinit) %initialize port record%
   PROCEDURE (po POINTER);                                              1j9a
      %declarations%                                                    1j9a1
         LOCAL ecb REF;                                                 1j9a1a
      %initialize send ECB%                                             1j9a2
         po.posndecb _ &ecb _ po + por.SIZE;                            1j9a2a
```

```
        ecb _ 1B6;                                                  1j9a2b
      %initialize receive ECB%                                      1j9a3
        po.porcvecb _ &ecb _ &ecb + 2;                              1j9a3a
        ecb _ 1B6;                                                  1j9a3b
      %return%                                                      1j9a4
        RETURN;                                                     1j9a4a
        END,                                                        1j9a4b
    %no termination%                                                1j9b
    %no handle transformation%                                      1j9c
    (poflsh) %flush port from system%
    PROCEDURE (poh);                                                1j9d
      %declarations%                                                1j9d1
        LOCAL port;                                                 1j9d1a
        LOCAL ps POINTER;                                           1j9d1b
      %disassociate port from process (if any)%                     1j9d2
        vwhlpls _ TRUE;                                             1j9d2a
        OPENPORT runfld (vpsfld, cshare : [port]);                  1j9d2b
        WHILE ps _ PCALL [port] (cprevue) DO                        1j9d2c
          IF ps.pspoh = poh AND PCALL [port] (clock) THEN           1j9d2c1
            BEGIN                                                   1j9d2c1a
            ps.pspoh _ 0;                                           1j9d2c1b
            EXIT LOOP;                                              1j9d2c1c
            END;                                                    1j9d2c1d
      %return%                                                      1j9d3
```

```
            RETURN;                                                 1j9d3a

            END,                                                    1j9d3b

%+PDP10% %processors%                                               1j10

   (prinit) %initialize processor record%
   PROCEDURE (pr POINTER);                                          1j10a

      %declarations%                                                1j10a1

         LOCAL ecb REF;                                             1j10a1a

      %initialize signin ECB%                                       1j10a2

         pr,prsiecb _ &ecb _ pr + prr,SIZE;                         1j10a2a

         ecb _ 1B6;                                                 1j10a2b

      %return%                                                      1j10a3

         RETURN;                                                    1j10a3a

         END,                                                       1j10a3b

   (prterm) %terminate processor record%
   PROCEDURE (pr POINTER);                                          1j10b

      %release attached storage%                                    1j10b1

         relent (cblock, pr,prstup);                                1j10b1a

      %return%                                                      1j10b2

         RETURN;                                                    1j10b2a

         END,                                                       1j10b2b

   (prhand) %make processor handle absolute%
   PROCEDURE (pcrh %-> pcrh%);                                      1j10c

      %declarations%                                                1j10c1

         LOCAL abspcrh;                                             1j10c1a

      %make processor handle absolute%                              1j10c2
```

```
        abspcrh _ CASE pcrh OF                                  1j10c2a
           = cfpcrh: vcpcrh;                                    1j10c2a1
           = clpcrh: vcspldr;                                   1j10c2a2
           ENDCASE   pcrh;                                      1j10c2a3
     %return%                                                   1j10c3
        RETURN (abspcrh);                                       1j10c3a
        END,                                                    1j10c3b

(prflsh) %flush processor from system%
PROCEDURE (pcrh);                                               1j10d
   %declarations%                                               1j10d1
      LOCAL port, nxtlsh, nxtpcrh, ldsc=0;                      1j10d1a
      LOCAL pcrid REF, lckid REF;                               1j10d1b
      LOCAL ls POINTER, pr POINTER;                             1j10d1c
   %unlock locks set by processor%                              1j10d2
      vwhlpls _ TRUE;                                           1j10d2a
      OPENPORT runfld (vlsfld, cdelete : [port]);               1j10d2b
      WHILE ls _ PCALL [port] (cprevue : nxtlsh) DO             1j10d2c
         IF ls.lshandle                                         1j10d2c1
         AND (&pcrid _ ls.lspcrid)                              1j10d2c2
         AND pcrid = vfph                                       1j10d2c3
         AND pcrid [2] = pcrh THEN                              1j10d2c4
            BEGIN                                               1j10d2c4a
            &lckid _ ls.lslckid;                                1j10d2c4b
            ldsc.ldlcb _ lckid;                                 1j10d2c4c
            ldsc.ldlsh _ nxtlsh;                                1j10d2c4d
```

```
                remlcb (ldsc);                                  1j10d2c4e
                END;                                            1j10d2c4f
        %delete/separate dependent processors%                  1j10d3
            OPENPORT runfld (vprfld, cdelete : [port]);         1j10d3a
            WHILE pr _ PCALL [port] (cprevue : nxtpcrh) DO      1j10d3b
                IF pr.prsuper = pcrh THEN                       1j10d3b1
                    IF pr.prsecond THEN sepfk (nxtpcrh)         1j10d3b1a
                    ELSE delpr (nxtpcrh, FALSE);                1j10d3b1b
        %return%                                                1j10d4
            RETURN;                                             1j10d4a
            END,%+PDP10%                                        1j10d4b
    %processes%                                                 1j11
        %no initialization%
                                                                1j11a
        (psterm) %terminate process record%
        PROCEDURE (ps POINTER);                                 1j11b
            %release attached storage%                          1j11b1
                relent (cblock, ps.pspsdesc);                   1j11b1a
            %return%                                            1j11b2
                RETURN;                                         1j11b2a
                END,
                                                                1j11b2b
        (pshand) %make process handle absolute%
        PROCEDURE (ph %-> ph%);                                 1j11c
            %declarations%                                      1j11c1
                LOCAL absph;                                    1j11c1a
            %make process handle absolute%                      1j11c2
```

```
        absph _ CASE ph OF                                          1j11c2a
           = cfph: vfph;                                            1j11c2a1
           = csph: vsph;                                            1j11c2a2
           ENDCASE ph;                                              1j11c2a3
     %return%                                                       1j11c3
        RETURN (absph);                                             1j11c3a
        END,                                                        1j11c3b

  (psflsh) %flush process from system%
  PROCEDURE (ph);                                                   1j11d
     %declarations%                                                 1j11d1
        LOCAL                                                       1j11d1a
           port, nxtph, nxtch, nxtsh, nxtpkh, sglocked,
           cnlocked;                                                1j11d1a1
        LOCAL                                                       1j11d1b
           ps POINTER, ca POINTER, sg POINTER, cn POINTER, pk
           POINTER;                                                 1j11d1b1
     %flush dependent introduced processes%                         1j11d2
        vwhlpls _ TRUE;                                             1j11d2a
        OPENPORT runfld (vpsfld, cdelete : [port]);                 1j11d2b
        WHILE ps _ PCALL [port] (cprevue : nxtph) DO                1j11d2c
           IF ps.pstype = cintroduced                               1j11d2c1
           AND ps.psadjph = ph                                      1j11d2c2
           AND PCALL [port] (clock) THEN                            1j11d2c3
              xdelchh (nxtph);                                      1j11d2c3a
     %flush dependent procedure calls%                              1j11d3
        OPENPORT runfld (vcafld, cdelete : [port]);                 1j11d3a
```

```
        WHILE ca _ PCALL [port] (cprevue : nxtch) DO          1j11d3b
          IF ca.caph = ph AND PCALL [port] (clock) THEN       1j11d3b1
            IF ca.caremote THEN relch (nxtch)                 1j11d3b1a
            ELSE xabrpe (ca,cacch);                           1j11d3b1b
      %close packages opened by process%                      1j11d4
        OPENPORT runfld (vpkfld, cdelete : [port]);           1j11d4a
        WHILE pk _ PCALL [port] (cprevue : nxtpkh) DO         1j11d4b
          IF rdrec (cph, pk) = ph                             1j11d4b1
          AND PCALL [port] (clock) THEN                       1j11d4b2
            iclspk (nxtpkh);                                  1j11d4b2a
      %delete/mark dead affected logical channels%            1j11d5
        OPENPORT runfld (vsgfld, cexclusive : [port]);        1j11d5a
        WHILE sg _ PCALL [port] (cprevue : nxtsh) DO          1j11d5b
          BEGIN                                               1j11d5b1
          sglocked _ FALSE;                                   1j11d5b2
          IF sg.sgph1 = ph                                    1j11d5b3
          AND sglocked _ PCALL [port] (clock) THEN            1j11d5b4
            sg.sg1dead _ TRUE;                                1j11d5b4a
          IF sg.sgph2 = ph                                    1j11d5b5
          AND (sglocked OR PCALL [port] (clock)) THEN         1j11d5b6
            IF sg.sgintro THEN sg.sg2dead _ TRUE              1j11d5b6a
            ELSE xdelchh (nxtsh);                             1j11d5b6b
          END;                                                1j11d5b7
      %mark dead affected physical channels%                  1j11d6
        OPENPORT runfld (vcnfld, cexclusive : [port]);        1j11d6a
```

```
            WHILE ca _ PCALL [port] (cprevue) DO                    1j11d6b
               BEGIN                                                1j11d6b1
               cnlocked _ FALSE;                                    1j11d6b2
               IF cn.cnph1 = ph                                     1j11d6b3
               AND cnlocked _ PCALL [port] (clock) THEN             1j11d6b4
                  cn.cn1dead _ TRUE;                                1j11d6b4a
               IF cn.cnph2 = ph                                     1j11d6b5
               AND (cnlocked OR PCALL [port] (clock)) THEN          1j11d6b6
                  cn.cn2dead _ TRUE;                                1j11d6b6a
               END;                                                 1j11d6b7
         %return%                                                   1j11d7
            RETURN;                                                 1j11d7a
            END.
                                                                    1j11d7b
%+PDP10% %segments%                                                 1j12
   %no initialization%
                                                                    1j12a
   %no termination%
                                                                    1j12b
   %no handle transformation%
                                                                    1j12c
   %no flush%%+PDP10%
                                                                    1j12d
%subprocesses%                                                      1j13
   (suinit) %initialize subprocess record%
   PROCEDURE (su POINTER);                                          1j13a
      %declarations%                                                1j13a1
         LOCAL ecb REF;                                             1j13a1a
```

```
    %initialize communication ECB%                                  1j13a2
      su.suwtecbs _ &ecb _ su + sur.SIZE;                           1j13a2a
      ecb _ (cmgmxh-cmgmnh+1)*1B6;                                  1j13a2b
    %return%                                                        1j13a3
      RETURN;                                                       1j13a3a
      END,
                                                                    1j13a3b

(suterm) %terminate subprocess record%
PROCEDURE (su POINTER);                                             1j13b
    %release attached storage%                                      1j13b1
      relent (ccharstr, su.susuaddr);                               1j13b1a
      relent (clist,    su.supknames);                              1j13b1b
      relent (clist,    su.supkhashs);                              1j13b1c
      relent (clist,    su.supkuses);                               1j13b1d
    %return%                                                        1j13b2
      RETURN;                                                       1j13b2a
      END,
                                                                    1j13b2b

(suhand) %make subprocess handle absolute%
PROCEDURE (sph %-> sph%);                                           1j13c
    %declarations%                                                  1j13c1
      LOCAL abssph;                                                 1j13c1a
    %make subprocess handle absolute%                               1j13c2
      abssph _ CASE sph OF                                          1j13c2a
        = cfsph: vcsph;                                             1j13c2a1
        = clsph: vpsldr;                                            1j13c2a2
        ENDCASE  sph;                                               1j13c2a3
```

```
        %return%                                                    1j13c3
          RETURN (abssph);                                          1j13c3a
          END,                                                      1j13c3b

      (suflsh) %flush subprocess from system%
      PROCEDURE (sph);                                              1j13d
        %declarations%                                              1j13d1
          LOCAL port;                                               1j13d1a
          LOCAL pk POINTER;                                         1j13d1b
        %mark affected packages dead%                               1j13d2
          vwhlpls _ TRUE;                                           1j13d2a
          OPENPORT runfld (vpkfld, cexclusive : [port]);            1j13d2b
          WHILE pk _ PCALL [port] (cprevue) DO                      1j13d2c
            IF pk.pksph = sph AND PCALL [port] (clock) THEN         1j13d2c1
              pk.pkdead _ TRUE;                                     1j13d2c1a
        %return%                                                    1j13d3
          RETURN;                                                   1j13d3a
          END,                                                      1j13d3b

    %system calls%                                                  1j14
      %no initialization%                                           1j14a

      (syterm) %terminate system call record%
      PROCEDURE (sy POINTER);                                       1j14b
        %release attached storage%                                  1j14b1
          relent (clist, sy.syargl);                                1j14b1a
          relent (clist, sy.syresl);                                1j14b1b
```

```
        %return%                                                    1j14b2
           RETURN;                                                  1j14b2a
           END,                                                     1j14b2b
     %no handle transformation%                                     1j14c
     %no flush%                                                     1j14d
  %user calls%                                                      1j15
     (usinit) %initialize user call record%
     PROCEDURE (us POINTER);                                        1j15a
        %declarations%                                              1j15a1
           LOCAL ecb REF;                                           1j15a1a
        %initialize ECB%                                            1j15a2
           us,usecb _ &ecb _ us + usr,SIZE;                         1j15a2a
           ecb _ 1B6;                                               1j15a2b
        %return%                                                    1j15a3
           RETURN;                                                  1j15a3a
           END,                                                     1j15a3b
     (usterm) %terminate user call record%
     PROCEDURE (us POINTER);                                        1j15b
        %release attached storage%                                  1j15b1
           relent (ccharstr, us,usdgmsg);                           1j15b1a
        %return%                                                    1j15b2
           RETURN;                                                  1j15b2a
           END,                                                     1j15b2b
```

```
        %no handle transformation%                                    1j15c
        %no flush%                                                    1j15d
      %timers%                                                        1j16
        %no initialization%                                           1j16a
        %no termination%                                              1j16b
        %no handle transformation%                                    1j16c
        %no flush%                                                    1j16d
  FINISH                                                              1k
FILE dpsosi % compiler (x110,)  output to (dpsosi.rel,) %             2
  %pre-compile switches%                                              2a
    ALLOW!;                                                           2a1
    NOMESS;                                                           2a2
    SET PDP10 = TRUE;                                                 2a3
  %declarations%                                                      2b
    %local pointers% POINTER                                          2b1
      vnxtmg,  %addr of nxt manager's record%                         2b1a
      vmg,     %addr of cur manager's record%                         2b1b
      vnewmg;  %addr of new manager's record%                         2b1c
    %global record definitions%                                       2b2
      (wdr) RECORD %inter-fork IPC window header%                     2b2a
        wdstate   [WORD], %window state%                              2b2a1
        wdmsgl [ADDRESS], %message length%                            2b2a2
```

```
        wdpktl [ADDRESS]; %packet length%                          2b2a3
      (adr) RECORD %address%                                       2b2b
        adofst       [9], %offset from page%                       2b2b1
        adpage       [9]; %page number%                            2b2b2
    %global constants%                                             2b3
      DECLARE EXTERNAL CONSTANT                                    2b3a
      %address space assignments%                                  2b3b
        %L10 runtime state%                                        2b3b1
          cpl10r  =   0B, %base page number%                       2b3b1a
          cll10r  =  13B, %length in pages%                        2b3b1b
        %managed storage%                                          2b3b2
          cpstrg  = 200B, %base page number%                       2b3b2a
          clstrg  = 100B, %length in pages%                        2b3b2b
        %processor window%                                         2b3b3
          cpprwn  = 300B, %base page number%                       2b3b3a
          clprwn  = 100B, %length in pages%                        2b3b3b
        %port window pairs%                                        2b3b4
          cppown  = 400B, %base page number of first%              2b3b4a
          clpown  =  10B, %length in pages of each pair%           2b3b4b
          cnpown  =  10B, %max number of pairs%                    2b3b4c
      %IPC parameters%                                             2b3c
        cintrfrk =   1, %inter-fork%                               2b3c1
        cintrhst =   2, %inter-host%                               2b3c2
        cintrjob =   3, %inter-job%                                2b3c3
        cmipctype=   3, %max IPC type%                             2b3c4
```

```
        cb36     =   1, %36-bit binary format%                          2b3c5
    %miscellaneous%                                                     2b3d
        cmchno   =  35, %max processor PSI channel number%              2b3d1
        cfkself  = 4B5; %self fork%                                     2b3d2
        DECLARE CONSTANT                                                2b3d3
        ctrerrs  = 2B8, %errors%                                        2b3d4
        cblock   =   1, %block%                                         2b3d5
        cpglen   = 512, %length (in words) of page%                     2b3d6
    %error numbers%                                                     2b3e
        eefimp   =  51, %not implemented%                               2b3e1
        eefops   =  52, %operating system error%                        2b3e2
        eemerr   =  53, %unidentifiable operating system error%         2b3e3
        ecwpkl   = 204, %inconsistent packet length%                    2b3e4
        esuser   = 408, %undefined user name%                           2b3e5
        erupsi   = 461, %undefined PSI channel number%                  2b3e6
        emient   = 502; %zero/negative entity size%                     2b3e7
%local constants% DECLARE CONSTANT                                      2b4
    %pseudo interrupt channel assignments%                              2b4a
        cpochn  =   1, %port communication%                             2b4a1
        cprchn  =   2, %processor communication%                        2b4a2
        ctmchn  =   3, %interval timer (OINITMR)%                       2b4a3
        ciichn  =  15, %illegal instruction%                            2b4a4
        cftchn  =  19, %fork termination%                               2b4a5
        cchprio =   1, %priority%                                       2b4a6
    %operating system characteristics%                                  2b4b
```

```
        cmhsnl  =  40, %max length of host name%                    2b4b1
        cmdirl  =  40, %max length of directory name%               2b4b2
        cmfnl   =  40, %max length of filename%                     2b4b3
        cmextl  =  40, %max length of extension%                    2b4b4
        cmpswl  =  40, %max length of password%                     2b4b5
      %inter-fork window states%                                    2b4c
        cidle   =   1, %window idle%                                2b4c1
        csent   =   2, %message sent%                               2b4c2
        crcvd   =   3, %message received%                           2b4c3
        cackd   =   4, %message acknowledged%                       2b4c4
      %miscellaneous%                                               2b4d
        cmcseml = 100, %max length of diagnostic%                   2b4d1
        cmprtl  = 200, %max length of print text%                   2b4d2
        cmnstg  =   4, %min length of zone segment%                 2b4d3
        cmxstg  =2048, %max length of zone segment%                 2b4d4
        cwtxtlen= (clpown*cPglen-wdr,SIZE)/2;                       2b4d5
          %length (in words) of window IPC text%                    2b4d5a
    %global catchphrases%                                           2b5
      (preljfn) CATCHPHRASE;                                        2b5a
        IF abr () AND SKIP !rljfn (CATCHPARAM) THEN NULL;           2b5a1
%initialization/termination%                                        2c
    (oinios) %initialize operating system%
    PROCEDURE;                                                      2c1
      %declarations%                                                2c1a
        LOCAL random;                                               2c1a1
```

```
    LOCAL STRING filename [cmfnl+1];                                    2c1a2
    LOCAL defdir REF, defext REF;                                       2c1a3
%init storage allocator%                                                2c1b
    vstgzon _ lstzone _ cpstrg*cpglen;                                  2c1b1
    makezone (vstgzon, cmxstg, cmnstg, clstrg*cpglen);                  2c1b2
%init pseudo interrupt system%                                          2c1c
    %set PSI table addresses%                                           2c1c1
        vlevtab [cchprio+1] _ $vpsipc;                                  2c1c1a
        !sir (cfkself, $vlevtab*1B6+$vchntab);                          2c1c1b
    %initialize JSYS trap channel%                                      2c1c2
        IF NOT SKIP !tfork (4B10, cprchn*1B6) THEN                      2c1c2a
            osigerr (R1);                                               2c1c2a1
    %create timer fork%                                                 2c1c3
        IF NOT SKIP !cfork (44B10, $tmrbgn+2) THEN                      2c1c3a
            osigerr (R1)                                                2c1c3a1
        ELSE vtmrfkh _ R1;                                              2c1c3b
    %activate required PSI channels%                                    2c1c4
        oactchn (cpochn, $oposigh);                                     2c1c4a
        oactchn (cprchn, $oprsigh);                                     2c1c4b
        oactchn (ctmchn, $otmsigh);                                     2c1c4c
        oactchn (ciichn, $oiisigh);                                     2c1c4d
        oactchn (cftchn, $oftsigh);                                     2c1c4e
    %enable PSI system%                                                 2c1c5
        !eir (cfkself);                                                 2c1c5a
%fetch own host address and job number%                                 2c1d
```

```
    !gjinf ();                                                    2c1d1
    vfhost _ !sysgt (dlhostn);                                    2c1d2
    vfjob  _ R3;                                                  2c1d3
  %init long GTJFN table%                                         2c1e
    vgtjtbl      _ 1B11;                                          2c1e1
    vgtjtbl [1] _ 3777773777777B;                                 2c1e2
    %temp                                                         2c1e3
      &defdir _ aloent (ccharstr, 6);                             2c1e3a
        *defdir* _ "WHITE", 0;                                    2c1e3a1
          vgtjtbl [3] _ opntstr (&defdir);%                       2c1e3a1a
    &defext _ aloent (ccharstr, 4);                               2c1e4
      *defext* _ "SAV", 0;                                        2c1e4a
        vgtjtbl [5] _ opntstr (&defext);                          2c1e4a1
  %create and open L10 state file%                                2c1f
    %generate unique filename%                                    2c1f1
      random _ !gtad (); random,RH _ !time ();                    2c1f1a
      *filename* _ "DPS-STATE.", STRING (random), ";T", 0;        2c1f1b
    %create file%                                                 2c1f2
      IF NOT SKIP !gtjfn (400001B6, opntstr ($filename))
      THEN                                                        2c1f2a
        osigerr (R1);                                             2c1f2a1
      vl10jfn _ R1,RH;                                            2c1f2b
      INVOKE (preljfn,, vl10jfn);                                 2c1f2c
    %open file%                                                   2c1f3
      IF NOT SKIP !openf (vl10jfn, 44000034B4) THEN               2c1f3a
```

```
            osigerr (R1);                                   2c1f3a1
        %return%                                            2c1g
           RETURN;                                          2c1g1
        %pseudo interrupt handlers%                         2c1h
           %illegal instruction%                            2c1h1
              (oiisigh):                                    2c1h1a
              !MOVEM R1,vpsiacs;                            2c1h1b
              !MOVEI R1,oillins;                            2c1h1c
              !EXCH  R1,vpsipc;                             2c1h1d
              !MOVEM R1,villpc;                             2c1h1e
              !MOVE  R1,vpsiacs;                            2c1h1f
              !debrk ();                                    2c1h1g
              (oillins): osigerr (0);                       2c1h1h
           %interval timer expiration%                      2c1h2
              (otmsigh): !AOS vwtdcde;                      2c1h2a
           %fork termination%                               2c1h3
              (oftsigh): !AOS vwtdcde;                      2c1h3a
           %port%                                           2c1h4
              (oposigh): !AOS vwtdcde;                      2c1h4a
           %processor%                                      2c1h5
              (oprsigh): !AOS vwtdcde;                      2c1h5a
           %save ACs%                                       2c1h6
              !MOVEM R1,vpsiacs;                            2c1h6a
              !MOVEI R1,vpsiacs;                            2c1h6b
              !MOVEM R0,1(R1);                              2c1h6c
```

```
        !JSP   R0,osvacs;                                   2c1h6d
      %signal watchdog manager%                             2c1h7
        IF vwtdecb.L < vwtdecb.M THEN                       2c1h7a
          vwtdecb [vwtdecb.L _ vwtdecb.L + 1] _             2c1h7a1
            vwtdcde;                                        2c1h7a1a
        vwtdcde _ 0;                                        2c1h7b
      %activate DPS if waiting%                             2c1h8
        IF vwaitpc THEN vpsipc _ (vwaitpc := 0);            2c1h8a
      %restore ACs%                                         2c1h9
        !MOVEI R1,vpsiacs;                                  2c1h9a
        !JSP   R0,orsacs;                                   2c1h9b
        !MOVEI R1,vpsiacs;                                  2c1h9c
        !MOVE  R0,1(R1);                                    2c1h9d
        !MOVE  R1,vpsiacs;                                  2c1h9e
      %return%                                              2c1h10
        !debrk ();                                          2c1h10a
    %DPS primary entry point%                               2c1i
      %save startup ACs%                                    2c1i1
        (dps):                                              2c1i1a
        !MOVEM R1,vstupacs;                                 2c1i1b
        !MOVEI R1,vstupacs;                                 2c1i1c
        !MOVEM R0,1(R1);                                    2c1i1d
        !JSP   R0,osvacs;                                   2c1i1e
        vstupacs _ (vstupacs [1] := vstupacs);              2c1i1f
      %temp%                                                2c1i2
```

```
        IF NOT vstupacs THEN                                    2c1i2a
          vstupacs [1] _ 6173151B5; %"cli"%                     2c1i2a1
      %locate source of clean XL10DATA pages%                   2c1i3
        R3 _ cpl10r; R3.LH _ cfkself;                           2c1i3a
        FOR R4 _ 1 UP UNTIL > cl110r DO                         2c1i3b
          BEGIN                                                 2c1i3b1
          !rpacs (R3);                                          2c1i3b2
          vfsvhnd [R4-1] _                                      2c1i3b3
            IF R2 ,A 1B10 THEN !rmap (R3) ELSE 0;               2c1i3b3a
          BUMP R3;                                              2c1i3b4
          END;                                                  2c1i3b5
      %begin execution%                                         2c1i4
        startup _ $dpsmain;                                     2c1i4a
        recover _ $dpsrcvr;                                     2c1i4b
        GOTO l10start;                                          2c1i4c
    %save ACS%                                                  2c1j
      (osvacs): %R0: return PC; R1: dst addr%                   2c1j1
      !MOVEM R2,2(R1);                                          2c1j2
      !MOVEI R2,15(R1);                                         2c1j3
      !ADDI  R1,3;                                              2c1j4
      !HRLI  R1,3;                                              2c1j5
      !BLT   R1,(R2);                                           2c1j6
      !MOVE  R1,R0;                                             2c1j7
      !JRST  2,(R1);                                            2c1j8
    %restore ACS%                                               2c1k
```

```
        (orsacs): %R0: return PC; R1: src addr%                    2c1k1
          !ADDI  R1,2;                                             2c1k2
          !HRLZS 0,R1;                                             2c1k3
          !HRRI  R1,2;                                             2c1k4
          !BLT   R1,15;                                            2c1k5
          !MOVE  R1,R0;                                            2c1k6
          !JRST  2,(R1);                                           2c1k7
      %timer fork ACs%                                             2c11
        (tmrbgn):                                                  2c111
        !MOVE  R1,vtmrintv;                                        2c112
        !disms ();                                                 2c113
        !MOVEI R1,777777B;                                         2c114
        !MOVSI R2,40000B;                                          2c115
        !iic   ();                                                 2c116
        !haltf ();                                                 2c117
        END.
                                                                   2c118
    (otrmos) %terminate operating system%
    PROCEDURE;                                                     2c2
      %delete timer fork%                                          2c2a
        !kfork (vtmrfkh := 0);                                     2c2a1
      %return%                                                     2c2b
        RETURN;                                                    2c2b1
        END.
                                                                   2c2b2
  %inter-fork IPC%                                                 2d
```

```
(fcrtps) %create fork%
PROCEDURE (po POINTER, action, host REF, intrahostaddr REF,
userinfo REF, stupinfo REF, remloc REF %-> remloc REF%);                2d1

   %declarations%                                                       2d1a

      LOCAL fkh, acs;                                                   2d1a1

   %catchphrases%                                                       2d1b

      (pdelfk)   CATCHPHRASE; IF abr () THEN odelfk (fkh);              2d1b1

      (prelfkpo) CATCHPHRASE; IF abr () THEN frelpo (po);               2d1b2

      (pdelfke)  CATCHPHRASE; IF abr () THEN fdelce (po);               2d1b3

   %create inferior CF%                                                 2d1c

      po,pofkh _ fkh _ ocrtfk ($dpsavf);                                2d1c1

      INVOKE (pdelfk);                                                  2d1c2

   %connect new fork to USERINFO-specified directory%                   2d1d

      odirfk (                                                          2d1d1

         fkh, ELEM #userinfo# [1], ELEM #userinfo# [2]);                2d1d1a

   %allocate local port%                                                2d1e

      #remloc# _ vfhost, vfjob, 1;                                      2d1e1

      relent (cblock, falopo (po, &remloc));                            2d1e2

      INVOKE (prelfkpo);                                                2d1e3

   %create channel%                                                     2d1f

      fcrtce (po, vwinbas);                                             2d1f1

      INVOKE (pdelfke);                                                 2d1f2

   %store channel parms for inferior%                                   2d1g

      acs _ cintrfrk*1B6+cb36;                                          2d1g1

      owracs (fkh, $acs);                                               2d1g2

   %start inferior CF%                                                  2d1h
```

```
        obgnfk (fkh);                                              2d1h1
    %await remote window initialization%                           2d1i
        frcvch (po);                                               2d1i1
    %return%                                                       2d1j
        RETURN (&remloc);                                          2d1j1
        END,                                                       2d1j2

(fdelps) %delete fork%                                             2d2
PROCEDURE (po POINTER);
    %declarations%                                                 2d2a
        LOCAL fkh;                                                 2d2a1
    %half inferior CF%                                             2d2b
        oendfk (fkh = po.Pofkh);                                   2d2b1
    %delete channel%                                               2d2c
        fdelce (po);                                               2d2c1
    %release local port%                                           2d2d
        frelpo (po);                                               2d2d1
    %delete inferior CF%                                           2d2e
        odelfk (fkh);                                              2d2e1
    %return%                                                       2d2f
        RETURN;                                                    2d2f1
        END,                                                       2d2f2

(fcrtce) %create end of inter-fork channel%                        2d3
PROCEDURE (po POINTER, remport REF);
    %declarations%                                                 2d3a
        LOCAL locport;                                             2d3a1
```

```
    %NOP unless active process%                                          2d3b
      IF po.poactive THEN                                                2d3b1
        BEGIN                                                            2d3b1a
        %initialize window%                                              2d3b1b
          [po.posndwd].wdstate _ cidle;                                  2d3b1b1
          [po.porcvwd].wdstate _ cidle;                                  2d3b1b2
        %create window%                                                  2d3b1c
          ocnnfk (0, po.polwbase, po.pofkh, po.porwbase _
          decstruc (cindex, &remport, 0), clpown);                       2d3b1c1
        END;                                                             2d3b1d
    %return%                                                             2d3c
      RETURN;                                                            2d3c1
      END.
                                                                         2d3c2

(fdelce) %delete end of inter-fork channel%
PROCEDURE (po POINTER);                                                  2d4
    %NOP unless active process%                                          2d4a
      IF po.poactive THEN                                                2d4a1
        odcnfk (po.pofkh, po.porwbase, clpown);                          2d4a1a
    %return%                                                             2d4b
      RETURN;                                                            2d4b1
      END.
                                                                         2d4b2

(falopo) %allocate local inter-fork port%
PROCEDURE (po POINTER, remloc REF %-> locport REF%);                     2d5
    %declarations%                                                       2d5a
      LOCAL number, locport, active;                                     2d5a1
```

```
    %catchphrases%                                                    2d5b
      (prelbit) CATCHPHRASE; IF abr () THEN                           2d5b1
        orelbit ($vwnmp, cwnmpl, number);                             2d5b1a
    %allocate address space window%                                   2d5c
      number _ oalobit ($vwnmp, cwnmpl);                              2d5c1
      INVOKE (prelbit);                                               2d5c2
      po.polwbase _ locport _ cppown + clpown*(number-1);             2d5c3
      active _ po.poactive;                                           2d5c4
      po.posndwd _                                                    2d5c5
        cpglen*(locport + active*clpown/2);                           2d5c5a
      po.porcvwd _                                                    2d5c6
        cpglen*(locport + (NOT active)*clpown/2);                     2d5c6a
    %return%                                                          2d5d
      RETURN (encstruc (cindex, locport));                            2d5d1
      END.                                                            2d5d2

(frelpo) %release local port%
PROCEDURE (po POINTER);                                               2d6
    %release window%                                                  2d6a
      orelbit (                                                       2d6a1
        $vwnmp, cwnmpl, (po.polwbase-cppown)/clpown+1);               2d6a1a
    %return%                                                          2d6b
      RETURN;                                                         2d6b1
      END.                                                            2d6b2

(fsndch) %send data to fork%
PROCEDURE (po POINTER, block REF);                                    2d7
```

```
%declarations%                                                    2d7a
   LOCAL src, srclen;                                             2d7a1
   LOCAL wd POINTER;                                              2d7a2
%initialize source pointers%                                      2d7b
   wd _ po,posndwd;                                               2d7b1
   wd,wdmsgl _ srclen _                                           2d7b2
      IF src _ &block THEN                                        2d7b2a
          sizent (cblock, &block)                                 2d7b2a1
      ELSE 0;                                                     2d7b2b
%output block%                                                    2d7c
   DO                                                             2d7c1
      BEGIN                                                       2d7c1a
      %move packet into window%                                   2d7c1b
         oblkxfr (                                                2d7c1b1
            src := src + cwtxtlen,                                2d7c1b1a
            wd + wdr,SIZE,                                        2d7c1b1b
            wd,wdpktl _                                           2d7c1b1c
               MIN (srclen := srclen - cwtxtlen, cwtxtlen));
                                                                  2d7c1b1c1
      %signal remote process%                                     2d7c1c
         wd,wdstate _ csent;                                      2d7c1c1
         osigfk (po,pofkh, cpochn);                               2d7c1c2
      %wait for acknowledgment%                                   2d7c1d
         waicok (po,posndecb);                                    2d7c1d1
      END                                                         2d7c1e
   WHILE srclen > 0;                                              2d7c2
```

```
    %return%                                                    2d7d
      RETURN;                                                   2d7d1
      END,
                                                                2d7d2
(frcvch) %receive data from fork%
PROCEDURE (po POINTER %-> block REF%);                          2d8
  %declarations%                                                2d8a
    LOCAL ecb, dst, dstlen, pktlen, block=0;                    2d8a1
    LOCAL wd POINTER;                                           2d8a2
  %wait for message%                                            2d8b
    ecb _ po.porcvecb;                                          2d8b1
    wd _ po.porcvwd;                                            2d8b2
    IF wd.wdstate # crcvd THEN waicok (ecb);                    2d8b3
  %allocate space for data%                                     2d8c
    IF dstlen _ wd.wdmsgl THEN                                  2d8c1
      BEGIN                                                     2d8c1a
      block _ dst _ aloent (cblock, dstlen);                    2d8c1b
      INVOKE (pabrrb,, block);                                  2d8c1c
      END;                                                      2d8c1d
  %input message%                                               2d8d
    DO                                                          2d8d1
      BEGIN                                                     2d8d1a
      %move packet out of window%                               2d8d1b
        IF (pktlen _ wd.wdpktl) > (dstlen := dstlen -
        pktlen) THEN                                            2d8d1b1
          sigerr (ecwpkl);                                      2d8d1b1a
```

```
            oblkxfr (                                                    2d8d1b2
              wd + wdr.SIZE, dst := dst + pktlen, pktlen);               2d8d1b2a
          %acknowledge receipt%                                          2d8d1c
            wd.wdstate _ cackd;                                          2d8d1c1
            osigfk (po.pofkh, cpochn);                                   2d8d1c2
          END                                                            2d8d1d
        WHILE dstlen > 0 AND waicok (ecb);                               2d8d2
      %return%                                                           2d8e
        RETURN (block);                                                  2d8e1
        END.                                                             2d8e2

(fupdpo) %update fork state%
PROCEDURE (po POINTER %=> aliveornot%);                                  2d9
    %declarations%                                                       2d9a
      LOCAL wd POINTER;                                                  2d9a1
    %update send window state%                                           2d9b
      wd _ po.posndwd;                                                   2d9b1
      IF wd.wdstate = cackd AND wd.wdstate _ cidle THEN                  2d9b2
        sigecb (po.posndecb, cok);                                       2d9b2a
    %update receive window state%                                        2d9c
      wd _ po.porcvwd;                                                   2d9c1
      IF wd.wdstate = csent AND wd.wdstate _ crcvd THEN                  2d9c2
        sigecb (po.porcvecb, cok);                                       2d9c2a
    %return%                                                             2d9d
      RETURN (ostsfk (po.pofkh));                                        2d9d1
```

```
        END,                                                    2d9d2

%inter-host IPC%                                                2e

   (ncrtps) %create network process%
   PROCEDURE (po POINTER, action, host REF, intrahostaddr REF,
   userinfo REF, stupinfo REF, remloc REF %-> remloc REF%);     2e1

      %not implemented%                                         2e1a

         sigerr (eefimp);                                       2e1a1

      %return%                                                  2e1b

         RETURN;                                                2e1b1

         END,                                                   2e1b2

   (ndelps) %delete network process%
   PROCEDURE (po POINTER);                                      2e2

      %not implemented%                                         2e2a

         sigerr (eefimp);                                       2e2a1

      %return%                                                  2e2b

         RETURN;                                                2e2b1

         END,                                                   2e2b2

   (ncrtce) %create end of inter-network-process channel%
   PROCEDURE (po POINTER, remport REF);                         2e3

      %not implemented%                                         2e3a

         sigerr (eefimp);                                       2e3a1

      %return%                                                  2e3b

         RETURN;                                                2e3b1

         END,                                                   2e3b2
```

```
(ndelce) %delete end of inter-network-process channel%
PROCEDURE (po POINTER);                                          2e4

   %not implemented%                                             2e4a

      sigerr (eefimp);                                           2e4a1

   %return%                                                      2e4b

      RETURN;                                                    2e4b1

      END.
                                                                 2e4b2

(nalopo) %allocate local inter-network-process port%
PROCEDURE (po POINTER, remloc REF %-> locport REF%);             2e5

   %not implemented%                                             2e5a

      sigerr (eefimp);                                           2e5a1

   %return%                                                      2e5b

      RETURN;                                                    2e5b1

      END.
                                                                 2e5b2

(nrelpo) %release local inter-network-process port%
PROCEDURE (po POINTER);                                          2e6

   %not implemented%                                             2e6a

      sigerr (eefimp);                                           2e6a1

   %return%                                                      2e6b

      RETURN;                                                    2e6b1

      END.
                                                                 2e6b2

(nsndch) %send data to network process%
PROCEDURE (po POINTER, block REF);                               2e7

   %not implemented%                                             2e7a

      sigerr (eefimp);                                           2e7a1
```

```
        %return%                                                    2e7b
           RETURN;                                                  2e7b1
           END,                                                     2e7b2

    (nrcvch) %receive data from network process%
    PROCEDURE (po POINTER %-> block REF%);                          2e8
       %not implemented%                                            2e8a
          sigerr (eefimp);                                          2e8a1
       %return%                                                     2e8b
          RETURN;                                                   2e8b1
          END,                                                      2e8b2

    (nupdpo) %update network process state%
    PROCEDURE (po POINTER %-> aliveornot%);                         2e9
       %not implemented%                                            2e9a
          sigerr (eefimp);                                          2e9a1
       %return%                                                     2e9b
          RETURN;                                                   2e9b1
          END,                                                      2e9b2

%forks%                                                             2f

   (ocrtfk)  %create fork%
   PROCEDURE (filename REF %-> fkh%);                               2f1
      %declarations%                                                2f1a
         LOCAL jfn, fkh;                                            2f1a1
         LOCAL STRING intfile [cmfnl+1];                            2f1a2
      %catchphrases%                                                2f1b
         (pkfork) CATCHPHRASE; IF abr () THEN !kfork (fkh);         2f1b1
```

```
    %locate SAV file%                                                    2f1c

       IF NOT SKIP !gtjfn ($vgtjtbl, ooptstr (&filename,
       $intfile)) THEN                                                   2f1c1

          osigerr (R1);                                                  2f1c1a

       jfn _ R1,RH;                                                      2f1c2

       INVOKE (preljfn,, jfn);                                           2f1c3

    %create inferior fork%                                               2f1d

       IF NOT SKIP !cfork (0) THEN osigerr (R1);                         2f1d1

       fkh _ R1;                                                         2f1d2

       INVOKE (pkfork);                                                  2f1d3

    %map SAV file into fork%                                             2f1e

       R1 _ jfn; R1,LH _ fkh;                                            2f1e1

       !get ();                                                          2f1e2

    %return%                                                             2f1f

       RETURN (fkh);                                                     2f1f1

       END.                                                              2f1f2

(odelfk)  %delete fork%
PROCEDURE (fkh);                                                         2f2

   %kill fork%                                                           2f2a

      !kfork (fkh);                                                      2f2a1

   %return%                                                              2f2b

      RETURN;                                                            2f2b1

      END.                                                               2f2b2

(oitdfk) %introduce fork%
PROCEDURE (locfkh, targetfkh %-> fkh%);                                  2f3
```

```
    %fetch handle for target fork%                                  2f3a
       IF NOT SKIP !gfrkh (locfkh, targetfkh) THEN                  2f3a1
          osigerr (R1);                                             2f3a1a
    %return%                                                        2f3b
       RETURN (R1);                                                 2f3b1
       END.                                                         2f3b2

(osepfk) %separate fork%
PROCEDURE (fkh);                                                    2f4
    %release fork handle%                                           2f4a
       !rfrkh (fkh);                                                2f4a1
    %return%                                                        2f4b
       RETURN;                                                      2f4b1
       END.                                                         2f4b2

(obgnfk)  %begin fork%
PROCEDURE (fkh);                                                    2f5
    %start fork%                                                    2f5a
       !sfrkv (fkh, 0);                                             2f5a1
    %return%                                                        2f5b
       RETURN;                                                      2f5b1
       END.                                                         2f5b2

(oendfk)  %end fork%
PROCEDURE (fkh);                                                    2f6
    %stop fork%                                                     2f6a
       IF fkh THEN !hfork (fkh)                                     2f6a1
       ELSE !haltf ();                                              2f6a2
```

```
    %return%                                                    2f6b
       RETURN;                                                  2f6b1
       END,
                                                                2f6b2
(omovfk)  %move fork%
PROCEDURE (srcfkh, dstfkh);                                     2f7
   %move fork%                                                  2f7a
      IF NOT SKIP !splfk (dstfkh, srcfkh) THEN                  2f7a1
         sigerr (R1);                                           2f7a1a
      !rfork (srcfkh);                                          2f7a2
   %return%                                                     2f7b
      RETURN;                                                   2f7b1
      END,
                                                                2f7b2
(ocnnfk)  %interconnect address spaces of two forks/files%
PROCEDURE (mainfkh, mainpg, auxfkh, auxpg, pgcnt %-> pgcnt%);   2f8
   %declarations%                                               2f8a
      LOCAL i;                                                  2f8a1
   %interconect address spaces%                                 2f8b
      R1    _ mainpg;                                           2f8b1
      R1,LH _ IF mainfkh THEN mainfkh ELSE cfkself;             2f8b2
      R2    _ auxpg;                                            2f8b3
      R2,LH _ IF  auxfkh THEN  auxfkh ELSE cfkself;             2f8b4
      R3    _ 16B10;                                            2f8b5
      FOR i _ 1 UP UNTIL > pgcnt DO                             2f8b6
         BEGIN                                                  2f8b6a
         !pmap ();                                              2f8b6b
```

```
            BUMP R1;                                                    2f8b6c
            BUMP R2;                                                    2f8b6d
            END;                                                        2f8b6e
        %return%                                                        2f8c
            RETURN (pgcnt);                                             2f8c1
            END,
                                                                        2f8c2
    (odcnfk)  %disconnect address spaces of two forks/files%
    PROCEDURE (auxfkh, auxpg, pgcnt);                                   2f9
        %declarations%                                                  2f9a
            LOCAL i;                                                    2f9a1
        %disconnect address spaces%                                     2f9b
            R1    _ -1;                                                 2f9b1
            R2    _ auxpg;                                              2f9b2
            R2,LH _ IF auxfkh THEN auxfkh ELSE cfkself;                 2f9b3
            FOR i _ 1 UP UNTIL > pgcnt DO                               2f9b4
                BEGIN                                                   2f9b4a
                !pmap ();                                               2f9b4b
                BUMP R2;                                                2f9b4c
                END;                                                    2f9b4d
        %return%                                                        2f9c
            RETURN;                                                     2f9c1
            END,
                                                                        2f9c2
    (ordacs)  %read fork's ACs%
    PROCEDURE (fkh, dst REF);                                           2f10
        %read fork's ACs%                                               2f10a
```

```
        !rfacs (fkh, &dst);                                    2f10a1
    %return%                                                   2f10b
        RETURN;                                                2f10b1
        END.
                                                               2f10b2
(owracs)  %write fork's ACs%
PROCEDURE (fkh, src REF);                                      2f11
    %write fork's ACs%                                         2f11a
        !sfacs (fkh, &src);                                    2f11a1
    %return%                                                   2f11b
        RETURN;                                                2f11b1
        END.
                                                               2f11b2
(otrpfk)  %intercept DPS operations%
PROCEDURE (fkh);                                               2f12
    %trap DPS JSYS%                                            2f12a
        IF NOT SKIP !tfork (4B11+fkh, $dtrptbl) THEN           2f12a1
            osigerr (R1);                                      2f12a1a
    %return%                                                   2f12b
        RETURN;                                                2f12b1
        END.
                                                               2f12b2
(outrfk)  %release DPS operations%
PROCEDURE (fkh);                                               2f13
    %untrap DPS JSYS%                                          2f13a
        IF NOT SKIP !tfork (2B11+fkh) THEN                     2f13a1
            osigerr (R1);                                      2f13a1a
    %return%                                                   2f13b
```

```
        RETURN;                                                 2f13b1

        END.
                                                                2f13b2

(ofndfk)  %identify fork initiating DPS operation%
PROCEDURE %(-> fkh)%;                                           2f14

   %read handle of trapped fork%                                2f14a

      IF NOT SKIP !rtfrk () THEN osigerr (R1);                  2f14a1

   %return%                                                     2f14b

      RETURN (R1,LH);                                           2f14b1

      END.
                                                                2f14b2

(ostsfk)  %return status of fork%
PROCEDURE (fkh %-> aliveornot%);                                2f15

   %read fork status%                                           2f15a

      !rfsts (fkh);                                             2f15a1

   %return%                                                     2f15b

      RETURN (R1,LH # 3);                                       2f15b1

      END.
                                                                2f15b2

(orsmfk)  %resume fork after DPS operation%
PROCEDURE (fkh, pcincr);                                        2f16

   %bump PC%                                                    2f16a

      IF pcincr >= 0 THEN                                       2f16a1

         BEGIN                                                  2f16a1a

         !rfsts (fkh);                                          2f16a1b

         !sfork (fkh, R2+pcincr);                               2f16a1c

         END;                                                   2f16a1d

   %untrap fork%                                                2f16b
```

```
        !utfrk ();                                              2f16b1

      %return%                                                  2f16c

        RETURN;                                                 2f16c1

        END.                                                    2f16c2

  (ocost)  %fetch current cost of fork%
  PROCEDURE (fkh %-> cost%);                                    2f17

    %fetch fork's runtime%                                      2f17a

      !runtm (fkh);                                             2f17a1

    %return%                                                    2f17b

      RETURN (R1);                                              2f17b1

      END.                                                      2f17b2

%signals%                                                       2g

  (owtsig)  %wait for signal%
  PROCEDURE;                                                    2g1

    %advance meter%                                             2g1a

      BUMP vmrwtc;                                              2g1a1

    %wait if necessary for PSI%                                 2g1b

      vwaitpc _ $owtrsm;                                        2g1b1

      IF NOT vwtdecb.L THEN                                     2g1b2

        BEGIN                                                   2g1b2a

        !JSYS 306B; %WAIT%                                      2g1b2b

        !JFCL; (owtrsm);                                        2g1b2c

        END;                                                    2g1b2d

    %return%                                                    2g1c

      RETURN;                                                   2g1c1
```

```
        END,                                                          2g1c2

(oactchn) %activate signal channel%
PROCEDURE (channel, haddr);                                           2g2

   %declarations%                                                     2g2a

      LOCAL mask;                                                     2g2a1

   %construct channel mask%                                           2g2b

      mask _ ochnmsk (channel);                                       2g2b1

   %update channel table%                                             2g2c

      vchntab [channel] _ cchprio*1B6+haddr;                          2g2c1

   %activate channel%                                                 2g2d

      !aic (cfkself, mask);                                           2g2d1

   %return%                                                           2g2e

      RETURN;                                                         2g2e1

      END,                                                            2g2e2

(ochnmsk) %construct channel mask%
PROCEDURE (channel %-> mask%);                                        2g3

   %verify channel%                                                   2g3a

      IF NOT (channel IN [0, cmchno]) THEN sigerr (erupsi);           2g3a1

   %construct channel mask%                                           2g3b

      R1 _ 4B11;                                                      2g3b1

      R2 _ = channel;                                                 2g3b2

      !LSH R1,(R2);                                                   2g3b3

   %return%                                                           2g3c

      RETURN (R1);                                                    2g3c1
```

```
        END,                                                    2g3c2

    (osigfk)  %signal fork%
    PROCEDURE (fkh, channel);                                   2g4

      %signal fork%                                             2g4a

        !iic (fkh, ochnmsk (channel));                          2g4a1

      %return%                                                  2g4b

        RETURN;                                                 2g4b1

        END,                                                    2g4b2

%files%                                                         2h

    (odirfk)  %connect fork to directory%
    PROCEDURE (fkh, directory REF, password REF);               2h1

      %declarations%                                            2h1a

        LOCAL dirno;                                            2h1a1

        LOCAL STRING intdir [cmdirl+1], intpass [cmpswl+1];     2h1a2

      %locate directory%                                        2h1b

        !stdir (1, ooptstr (&directory, $intdir));              2h1b1

        GOTO odirfail;                                          2h1b2

        GOTO odirfail;                                          2h1b3

        dirno _ R1,RH;                                          2h1b4

      %connect fork to directory%                               2h1c

        IF NOT SKIP !cndir (4B11+dirno, ooptstr (&password,
        $intpass)) THEN                                         2h1c1

          osigerr (R1);                                         2h1c1a

      %return%                                                  2h1d

        RETURN;                                                 2h1d1
```

```
        %abort%                                                          2h1e
           (odirfail): sigerr (esuser);                                  2h1e1
           END,
                                                                         2h1e2
    %events%                                                             2i
       (otest) %test for signal%
       PROCEDURE (ecb REF, removeornot %-> code, newlength%);            2i1
          %declarations%                                                 2i1a
             LOCAL code;                                                 2i1a1
          %remove oldest code from ECB%                                  2i1b
             IF code _ ecb.L THEN                                        2i1b1
                BEGIN                                                    2i1b1a
                code _ ecb [1];                                          2i1b1b
                IF removeornot THEN                                      2i1b1c
                   BEGIN                                                 2i1b1c1
                   %disable interrupts%                                  2i1b1c2
                      !dir (cfkself);                                    2i1b1c2a
                   %remove oldest completion code%                       2i1b1c3
                      oblkxfr (&ecb+2, &ecb+1, (ecb.L _ ecb.L - 1));     2i1b1c3a
                   %enable interrupts%                                   2i1b1c4
                      !eir (cfkself);                                    2i1b1c4a
                   END;                                                  2i1b1c5
                END;                                                     2i1b1d
          %return%                                                       2i1c
             RETURN (code, ecb.L);                                       2i1c1
```

```
        END,                                                          2i1c2

    (owait)  %wait for any of a list of events%
    PROCEDURE (ecbs REF, listornot %-> code, index, length%);          2i2

      %update/delete current environment%                             2i2a

        IF &ecbs THEN                                                 2i2a1

          BEGIN                                                       2i2a1a

          vmg,mgecbs _ &ecbs;                                         2i2a1b

          vmg,mglon  _ listornot;                                     2i2a1c

          END                                                         2i2a1d

        ELSE delrec (vmg);                                            2i2a2

      %select next manager for dispatch%                              2i2b

        selmgr ();                                                    2i2b1

        IF &ecbs AND vmgh = vnewh THEN                                2i2b2

          RETURN (vcode, vindex, vlength);                            2i2b2a

        vnewacs  _ vnewmg,mgacs;                                      2i2b3

        vnewbase _ vnewmg,mgfwbase;                                   2i2b4

      %save current environment%                                      2i2c

        IF &ecbs THEN                                                 2i2c1

          BEGIN                                                       2i2c1a

          %DPS state%                                                 2i2c1b

            oblkxfr ($vstate, vmg,mgstate, $vstalen-$vstate);         2i2c1b1

          %L10 state%                                                 2i2c1c

            %automatic -- in file%                                    2i2c1c1

          %ACs%                                                       2i2c1d

            !MOVEM R1,@vacs;                                          2i2c1d1
```

```
            !MOVE  R1,vacs;                                    2i2c1d2
            !MOVEM R0,1(R1);                                   2i2c1d3
            !JSP   R0,osvacs;                                  2i2c1d4
         END;                                                  2i2c1e
      %restore next environment%                               2i2d
        vmg  _ vnewmg;                                         2i2d1
        vmgh _ vnewh;                                          2i2d2
        vacs _ vnewacs;                                        2i2d3
        IF vindex THEN                                         2i2d4
          BEGIN                                                2i2d4a
          %ACs%                                                2i2d4b
            !MOVE  R1,vacs;                                    2i2d4b1
            !JSP   R0,orsacs;                                  2i2d4b2
            !MOVE  R1,vacs;                                    2i2d4b3
            !MOVE  R0,1(R1);                                   2i2d4b4
            !MOVE  R1,@vacs;                                   2i2d4b5
          %L10 state%                                          2i2d4c
            R1 _ vnewbase; R1,LH _ vl10jfn;                    2i2d4c1
            R2    _ cpl10r;                                    2i2d4c2
            R2,LH _ cfkself;                                   2i2d4c3
            R3 _ 16B10;                                        2i2d4c4
            FOR R4 _ 1 UP UNTIL > cl110r DO                    2i2d4c5
              BEGIN                                            2i2d4c5a
              !pmap ();                                        2i2d4c5b
              BUMP R1;                                         2i2d4c5c
```

```
                BUMP R2;                                                2i2d4c5d
                END;                                                    2i2d4c5e
            %DPS state%                                                 2i2d4d
               oblkxfr (vmg,mgstate, svstate, svstalen-svstate);        2i2d4d1
            RETURN (vcode, vindex, vlength);                            2i2d4e
            END                                                         2i2d4f
        %create next environment%                                       2i2e
          ELSE                                                          2i2e1
            BEGIN                                                       2i2e1a
            %map in fresh copy of XL10DATA%                             2i2e1b
               R2 _ cpl10r; R2.LH _ cfkself;                            2i2e1b1
               R3 _ 1204B8;                                             2i2e1b2
               FOR R4 _ 1 UP UNTIL > cl110r DO                          2i2e1b3
                  BEGIN                                                 2i2e1b3a
                  R1 _                                                  2i2e1b3b
                     IF vfsvhnd [R4-1] THEN vfsvhnd [R4-1] ELSE
                     -1;                                                2i2e1b3b1
                  !pmap ();                                             2i2e1b3c
                  BUMP R2;                                              2i2e1b3d
                  END;                                                  2i2e1b3e
            %zero DPS state%                                            2i2e1c
               oblkxfr (vmg,mgstate, svstate, svstalen-svstate);        2i2e1c1
               verrmsg.M _ cmerml;                                      2i2e1c2
            %enter L10%                                                 2i2e1d
               startup _ $mgcover;                                      2i2e1d1
```

```
                GOTO l10start;                                  2i2e1d2
              END;                                              2i2e1e
        %return%                                                2i2f
           END.
                                                                2i2f1
  %timer%                                                       2j
     (osettmr) %set interval timer%
     PROCEDURE (interval);                                      2j1
        %set interval timer%                                    2j1a
           vtmrintv _ interval;                                 2j1a1
           vtmrstrt _ !time ();                                 2j1a2
        %arrange for signal on expiration%                      2j1b
           !sfork (vtmrfkh, 2);                                 2j1b1
        %return%                                                2j1c
           RETURN;                                              2j1c1
           END.
                                                                2j1c2
     (otsttmr) %test/cancel interval timer%
     PROCEDURE (cancel %-> gone, left%);                        2j2
        %declarations%                                          2j2a
           LOCAL gone, left;                                    2j2a1
        %test interval timer%                                   2j2b
           gone _ IF vtmrintv THEN !time () - vtmrstrt ELSE 0;  2j2b1
           left _ vtmrintv - gone;                              2j2b2
        %cancel timer if necessary%                             2j2c
           IF cancel THEN                                       2j2c1
              BEGIN                                             2j2c1a
```

```
            !hfork (vtmrfkh);                                   2j2c1b
            vtmrintv _ 0;                                       2j2c1c
            END;                                                2j2c1d
        %return%                                                2j2d
            RETURN (gone, left);                                2j2d1
            END,
                                                                2j2d2
%storage%                                                       2k
    (oprtstr) %print string at user's terminal%
    PROCEDURE (text REF);                                       2k1
        %declarations%                                          2k1a
            LOCAL STRING inttext [cmprtl+1];                    2k1a1
        %NOP if no text%                                        2k1b
            IF NOT &text THEN RETURN;                           2k1b1
        %format text%                                           2k1c
            *inttext* _                                         2k1c1
                *vpsname*, " (", STRING (vmgh-cmgmnh+1), "): ",
                *text*, EOL, 0;                                 2k1c1a
        %output text%                                           2k1d
            !psout (opntstr ($inttext));                        2k1d1
        %return%                                                2k1e
            RETURN;                                             2k1e1
            END,
                                                                2k1e2
    (ostrxfr) %copy string%
    PROCEDURE (srcbp, dstbp, length);                           2k2
        %verify size of string%                                 2k2a
```

```
        IF length < 0 THEN sigerr (emient);                          2k2a1
    %copy string%                                                    2k2b
        IF length AND dstbp THEN                                     2k2b1
            !sin (srcbp, dstbp, -length);                            2k2b1a
    %return%                                                         2k2c
        RETURN;                                                      2k2c1
        END.                                                         2k2c2

(ostrsiz) %return length of ASCIZ string%
PROCEDURE (bp REF, boundary %-> length, overflow%);                  2k3
    %declarations%                                                   2k3a
        LOCAL overflow=FALSE;                                        2k3a1
    %count characters until NUL/boundary%                            2k3b
        R2 _ 0;                                                      2k3b1
        R3 _ &bp;                                                    2k3b2
        DO                                                           2k3b3
            BEGIN                                                    2k3b3a
            !ILDB R1,(R3);                                           2k3b3b
            IF R1 THEN BUMP R2;                                      2k3b3c
            END                                                      2k3b3d
        UNTIL NOT R1 OR overflow _ (R3,RH >= boundary,RH);           2k3b4
    %return%                                                         2k3c
        RETURN (R2, overflow);                                       2k3c1
        END.                                                         2k3c2

(oiptstr) %input ASCIZ to L10 string%
PROCEDURE (srcbp, dst REF %-> dst REF%);                             2k4
```

```
        %declarations%                                              2k4a
           LOCAL char;                                              2k4a1
        %append characters until NUL%                               2k4b
           dst.L _ 0;                                               2k4b1
           WHILE char _ ordbyt ($srcbp) DO                          2k4b2
              *dst* _ *dst*, char;                                  2k4b2a
        %return%                                                    2k4c
           RETURN (&dst);                                           2k4c1
           END.                                                     2k4c2

    (ooptstr) %output L10 to ASCIZ string%
    PROCEDURE (src REF, dst REF %-> bp%);                           2k5
        %construct ASCIZ equivalent of source string%               2k5a
           *dst* _ *src*, 0;                                        2k5a1
        %return%                                                    2k5b
           RETURN (opntstr (&dst));                                 2k5b1
           END.                                                     2k5b2

    (ordbyt)  %read byte%
    PROCEDURE (bp REF %-> byte%);                                   2k6
        %read next byte%                                            2k6a
           R1 _ &bp;                                                2k6a1
           !ILDB R2,(R1);                                           2k6a2
        %return%                                                    2k6b
           RETURN (R2);                                             2k6b1
           END.                                                     2k6b2
```

```
(opntstr) %return byte pointer to L10 string%                    2k7
PROCEDURE (src REF %-> bp%);

   %return%                                                      2k7a

      RETURN (opntwrd (&src+1, CHAR));                           2k7a1

      END.                                                       2k7a2

(opntwrd) %return byte pointer to word%                          2k8
PROCEDURE (addr, bytesize %-> bp%);

   %return%                                                      2k8a

      RETURN (MKFD (WORD, bytesize, addr));                      2k8a1

      END.                                                       2k8a2

(oblkxfr) %copy block%                                           2k9
PROCEDURE (src REF, dst REF, length);

   %verify size of block%                                        2k9a

      IF length < 0 THEN sigerr (emient);                        2k9a1

   %copy block%                                                  2k9b

      IF length AND &dst THEN                                    2k9b1

         BEGIN                                                   2k9b1a

         R1 _ &dst; R1,LH _ &src;                                2k9b1b

         R2 _ &dst + length - 1;                                 2k9b1c

         !BLT R1,(R2);                                           2k9b1d

         END;                                                    2k9b1e

   %return%                                                      2k9c

      RETURN;                                                    2k9c1

      END.                                                       2k9c2
```

```
(oalobit) %allocate map bit%                                        2k10
PROCEDURE (map REF, length %-> number%);

   %declarations%                                                   2k10a

      LOCAL end=&map+length, count=0;                               2k10a1

   %find next unallocated bit%                                      2k10b

      UNTIL &map >= end DO                                          2k10b1

         BEGIN                                                      2k10b1a

         R1 _ map;                                                  2k10b1b

         !JFFO R1,pfound;                                           2k10b1c

         BUMP &map;                                                 2k10b1d

         count _ count + 36                                         2k10b1e

         END;                                                       2k10b1f

      RETURN (FALSE);                                               2k10b2

   %set bit allocated%                                              2k10c

      (pfound);                                                     2k10c1

      R4 _ 4B11;                                                    2k10c2

      R3 _ - R2;                                                    2k10c3

      !LSH R4,(R3);                                                 2k10c4

      !TDZ R1,R4;                                                   2k10c5

      map _ R1;                                                     2k10c6

   %return%                                                         2k10d

      RETURN (R2+count+1);                                          2k10d1

      END.
                                                                    2k10d2

(orelbit) %release map bit%
PROCEDURE (map REF, length, number);                                2k11
```

```
        %declarations%                                              2k11a
           LOCAL quo, rem;                                          2k11a1
        %set bit unallocated%                                       2k11b
           DIV number / 36, quo, rem;                               2k11b1
           &map _ &map + quo;                                       2k11b2
           R1 _ map;                                                2k11b3
           R3 _ 1 ^ rem;                                            2k11b4
           R4 _ 4B11;                                               2k11b5
           !LSH R4,(R3);                                            2k11b6
           !TDO R1,R4;                                              2k11b7
           map _ R1;                                                2k11b8
        %return%                                                    2k11c
           RETURN;                                                  2k11c1
           END,                                                     2k11c2

    %errors%                                                        21

       (osigerr) %Operating system error%
       PROCEDURE (number);                                          211

         %declarations%                                             211a
            LOCAL dst, len=0;                                       211a1
         %advance meter%                                            211b
            BUMP vmoerc;                                            211b1
         %fetch error message from operating system%                211c
            R2 _ IF number THEN number ELSE -1; R2,LH _ cfkself;    211c1
            R3 _ 0;  R3,LH _ -cmoseml;                              211c2
            !erstr (dst _ opntstr ($verrmsg));                      211c3
```

```
            GOTO osigfail;                                          211c4
            GOTO osigfail;                                          211c5
            WHILE ordbyt ($dst) DO BUMP len;                        211c6
            verrmsg,L _ len;                                        211c7
         %trace error%                                              211d
            IF dtrace ,A ctrerrs THEN                               211d1
               trcerr (eefops, $verrmsg);                           211d1a
         %abort%                                                    211e
            ABORT (eefops, $verrmsg);                               211e1
            (osigfail); sigerr (eemerr);                            211e2
            END.                                                    211e3
   FINISH                                                           2m
FILE dpstbl % compiler (x110,)  output to (dpstbl,rel,) %           3
   %pre-compile switches%                                           3a
      NOMESS;                                                       3a1
      SET PDP10 = TRUE;                                             3a2
   %declarations%                                                   3b
      %global constants%                                            3b1
         DECLARE EXTERNAL CONSTANT                                  3b1a
         %formal data types%                                        3b1b
            cempty   =   1, %empty%                                 3b1b1
            cboolean =   2, %boolean%                               3b1b2
            cindex   =   3, %index%                                 3b1b3
            cinteger =   4, %integer%                               3b1b4
```

```
        cbitstr  =   5, %bit string%                              3b1b5
        ccharstr =   6, %character string%                        3b1b6
        clist    =   7, %list%                                    3b1b7
        cmfdtn   =   7, %max formal data type number%             3b1b8
    %informal data types%                                         3b1c
        cmnlcdt  =   8, %min data type number in table%           3b1c1
        cdsel    =   8, %data store selector%                     3b1c2
        cintpsel =   9, %internal procedure selector%             3b1c3
        cextpsel =  10, %external procedure selector%             3b1c4
        cusrinf  =  11, %user information%                        3b1c5
        cpsloc   =  12, %process location%                        3b1c6
        cpsdesc  =  13, %process description%                     3b1c7
        cabrinfo =  14, %abort information%                       3b1c8
        cpkinfo  =  15, %package information%                     3b1c9
        cmxlcdt  =  15, %max data type number in table%           3b1c10
        %----------%                                             3b1c11
        cucstr   =  16, %upper-case character string%             3b1c12
        cmessage =  17, %IPC message%                             3b1c13
        cesel    =  18, %element selector%                        3b1c14
        cstruc   =  19, %arbitrary structure%                     3b1c15
        cdata    =  20, %data%                                    3b1c16
        csmlblk  =  21, %small block (RDPR)%                      3b1c17
        cdumemp  =  22, %dummy empty%                             3b1c18
        cdumlist =  23, %dummy list%                              3b1c19
        calostruc= 24, %allocated structure%                      3b1c20
```

```
        calodata =  25, %allocated data%                                3b1c21
    %handle bounds%                                                     3b1d
        cprmnh = 3, cprmxh = 34, %processor handles%                    3b1d1
            %note also the following relative handles%                  3b1d1a
            cfpcrh = 1, %self%                                          3b1d1b
            clpcrh = 2, %subprocess leader%                             3b1d1c
        cmgmnh =35, cmgmxh = 63, %manager handles%                      3b1d2
            %PCRHs and MGHs must be drawn from one name space%          3b1d2a
        cpsmnh = 3, cpsmxh = 34, %process handles%                      3b1d3
            %note also the following relative handles%                  3b1d3a
            cfph   = 1, %self%                                          3b1d3b
            csph   = 2, %direct superior%                               3b1d3c
        csgmnh =35, csgmxh = 63, %segment handles%                      3b1d4
            %PHs and SHs must be drawn from one name space%             3b1d4a
        csumnh = 3, csumxh = 63, %subprocess handles%                   3b1d5
            %note also the following relative handles%                  3b1d5a
            cfsph  = 1, %self%                                          3b1d5b
            clsph  = 2, %process leader%                                3b1d5c
        ccamnh = 1, ccamxh = 63, %call handles%                         3b1d6
        ccnmnh = 1, ccnmxh = 63, %channel handles%                      3b1d7
        cdtmnh = 1, cdtmxh = 63, %data store handles%                   3b1d8
        cevmnh = 1, cevmxh = 63, %event handles%                        3b1d9
        clkmnh = 1, clkmxh = 63, %lock handles%                         3b1d10
        clsmnh = 1, clsmxh = 63, %lock set handles%                     3b1d11
        cpkmnh = 1, cpkmxh = 63, %package handles%                      3b1d12
```

```
        cpomnh = 1, cpomxh = 63, %port handles%                    3b1d13
        csymnh = 1, csymxh = 63, %system call handles%             3b1d14
        cusmnh = 1, cusmxh = 63, %user call handles%               3b1d15
        ctmmnh = 1, ctmmxh = 63, %timer handles%                   3b1d16
    %error numbers%                                                3b1e
        %undefined handles%                                        3b1e1
            egmhca = 551, %undefined call handle%                  3b1e1a
            egmhcn = 552, %undefined channel handle%               3b1e1b
            egmhdt = 553, %undefined data store handle%            3b1e1c
            egmhev = 554, %undefined event handle%                 3b1e1d
            egmhlk = 555, %undefined lock handle%                  3b1e1e
            egmhls = 556, %undefined lockset handle%               3b1e1f
            egmhmg = 557, %undefined manager handle%               3b1e1g
            egmhpk = 558, %undefined package handle%               3b1e1h
            egmhpo = 559, %undefined port handle%                  3b1e1i
            egmhpr = 560, %undefined processor handle%             3b1e1j
            egmhps = 561, %undefined process handle%               3b1e1k
            egmhsg = 562, %undefined segment handle%               3b1e1l
            egmhsu = 563, %undefined subprocess handle%            3b1e1m
            egmhSy = 564, %undefined system call handle%           3b1e1n
            egmhus = 565, %undefined user call handle%             3b1e1o
            egmhtm = 566, %undefined timer handle%                 3b1e1p
        %no more handles%                                          3b1e2
            ehohca = 601, %no available call handle%               3b1e2a
            ehohcn = 602, %no available channel handle%            3b1e2b
```

```
        ehohdt = 603, %no available data store handle%          3b1e2c
        ehohev = 604, %no available event handle%               3b1e2d
        ehohlk = 605, %no available lock handle%                3b1e2e
        ehohls = 606, %no available lockset handle%             3b1e2f
        ehohmg = 607, %no available manager handle%             3b1e2g
        ehohpk = 608, %no available package handle%             3b1e2h
        ehohpo = 609, %no available port handle%                3b1e2i
        ehohpr = 610, %no available processor handle%           3b1e2j
        ehohps = 611, %no available process handle%             3b1e2k
        ehohsg = 612, %no available segment handle%             3b1e2l
        ehohsu = 613, %no available subprocess handle%          3b1e2m
        ehohsy = 614, %no available system call handle%         3b1e2n
        ehohus = 615, %no available user call handle%           3b1e2o
        ehohtm = 616, %no available timer handle%               3b1e2p
    %processor intra-AC parameter locations%                    3b1f
      cno = 0, %nowhere%                                        3b1f1
      crh = 0, %right half%                                     3b1f2
      clh = 1, %left half%                                      3b1f3
      cwh = 2, %whole AC%                                       3b1f4
      cfg = 3, %LH to LH (flags)%                               3b1f5
    %miscellaneous%                                             3b1g
      cjuprio   =   1, %default JUSR priority%                  3b1g1
      cmfldn    =  16, %number of folders%                      3b1g2
      cprpmspcl =  12; %length (in bits) of parm spec%          3b1g3
      DECLARE CONSTANT                                          3b1g4
```

```
        chndwid   =   6, %width of local handle%                    3b1g5
        cmlcbl    =   3, %length of LCB%                            3b1g6
        cmlkidl   =   3, %length of lock identifier%                3b1g7
        cmpidl    =   3, %length of processor identifier%           3b1g8
        cnoacs    =  16; %number of ACs (see also DPSDTA)%          3b1g9
    %local constants% DECLARE CONSTANT                              3b2
      %table offsets%                                               3b2a
        %user/system calls%                                         3b2a1
          cargs   = 1B3, %argument count%                           3b2a1a
          cress   = 1B0, %result count%                             3b2a1b
          cacn    = 1B3, %AC-1%                                     3b2a1c
          cpos    = 2B2, %intra-AC position%                        3b2a1d
          cempok  = 4B1, %empty-ok%                                 3b2a1e
          clistof = 1B2, %list-of%                                  3b2a1f
          carg1   = 1B8, %argument 1%                               3b2a1g
          carg2   = 1B4, %argument 2%                               3b2a1h
          carg3   = 1B0, %argument 3%                               3b2a1i
          carg4   = 1B8, %argument 4%                               3b2a1j
          carg5   = 1B4, %argument 5%                               3b2a1k
          carg6   = 1B0, %argument 6%                               3b2a1l
          carg7   = 1B8, %argument 7%                               3b2a1m
          carg8   = 1B4, %argument 8%                               3b2a1n
          cres1   = 1B8, %result 1%                                 3b2a1o
          cres2   = 1B4, %result 2%                                 3b2a1p
          cres3   = 1B0, %result 3%                                 3b2a1q
```

```
%messages%                                                          3b2a2
    cprms   = 1B3, %parameter count%                                3b2a2a
    cprm1   = 1B8, %parameter 1%                                    3b2a2b
    cprm2   = 1B4, %parameter 2%                                    3b2a2c
    cprm3   = 1B0, %parameter 3%                                    3b2a2d
    cprm4   = 1B8, %parameter 4%                                    3b2a2e
    cprm5   = 1B4, %parameter 5%                                    3b2a2f
    cprm6   = 1B0, %parameter 6%                                    3b2a2g
    cprm7   = 1B8, %parameter 7%                                    3b2a2h
    cprm8   = 1B4, %parameter 8%                                    3b2a2i
    cprm9   = 1B0, %parameter 9%                                    3b2a2j
    cprm10  = 1B8, %parameter 10%                                   3b2a2k
    cprm11  = 1B4, %parameter 11%                                   3b2a2l
    cprm12  = 1B0, %parameter 12%                                   3b2a2m
%data structures%                                                   3b2a3
    celms   = 1B3, %element count%                                  3b2a3a
    celm1   = 1B8, %element 1%                                      3b2a3b
    celm2   = 1B4, %element 2%                                      3b2a3c
    celm3   = 1B0, %element 3%                                      3b2a3d
    celm4   = 1B8, %element 4%                                      3b2a3e
    celm5   = 1B4, %element 5%                                      3b2a3f
    celm6   = 1B0, %element 6%                                      3b2a3g
%folders%                                                           3b2a4
    crcwl   = 1B0, %length (in words) of record workspace%         3b2a4a
    cmnhd   = 1B2, %min handle%                                     3b2a4b
```

```
            cmxhd   = 1B4, %max handle%                                  3b2a4c
            cennex  = 1B0, %error number for non-existent handle%        3b2a4d
            cenovf  = 1B4, %error number for folder overflow%            3b2a4e
            cmapl   = 1B8, %length (in words) of handle map%             3b2a4f
        %miscellaneous%                                                  3b2b
           cmptchl = 100; %length of patch space%                        3b2b1
    %-------------------------------------------------%                  3c
    %   ...a macro, a macro, my kingdom for a macro...   %               3d
    %-------------------------------------------------%                  3e
    %folder definition table%                                            3f
       %model                                                            3f1
          fldadr, recdef, init, term, hand,                              3f1a
          wslen*crcwl + minhnd*cmnhd + maxhnd*cmxhd,                     3f1b
          nexerr*cennex + ovferr*cenovf + maplen*cmapl, %
                                                                         3f1c
       %calls%                                                           3f2
          DECLARE EXTERNAL dfldtbl = (                                   3f2a
          $vcafld, $car, $cainit, $caterm, 0,                            3f2b
          4*crcwl + ccamnh*cmnhd + ccamxh*cmxhd,                         3f2c
          egmhca*cennex + ehohca*cenovf                                  3f2d
             + 2*cmapl);
                                                                         3f2d1
       %+PDP10% %channels%                                               3f3
          DECLARE dcndef = (                                             3f3a
          $vcnfld, $cnr, 0, 0, 0,                                        3f3b
          ccnmnh*cmnhd + ccnmxh*cmxhd,                                   3f3c
```

```
        egmhcn*cennex + ehohcn*cenovf                                   3f3d
           + 2*cmapl);%+PDP10%
                                                                        3f3d1
    %+PDP10% %stores%                                                   3f4
        DECLARE ddtdef = (                                              3f4a
        $vdtfld, $dtr, $dtinit, $dtterm, 0,                             3f4b
        (cmlcbl+cmpidl)*crcwl + cdtmnh*cmnhd + cdtmxh*cmxhd,            3f4c
        egmhdt*cennex + ehohdt*cenovf                                   3f4d
           + 2*cmapl);%+PDP10%
                                                                        3f4d1
    %+PDP10% %events%                                                   3f5
        DECLARE devdef = (                                              3f5a
        $vevfld, $evr, 0, $evterm, 0,                                   3f5b
        cevmnh*cmnhd + cevmxh*cmxhd,                                    3f5c
        egmhev*cennex + ehohev*cenovf                                   3f5d
           + 2*cmapl);%+PDP10%
                                                                        3f5d1
    %locks%                                                             3f6
        DECLARE dlkdef = (                                              3f6a
        $vlkfld, $lkr, $lkinit, $lkterm, 0,                             3f6b
        (cmlcbl+cmpidl)*crcwl + clkmnh*cmnhd + clkmxh*cmxhd,            3f6c
        egmhlk*cennex + ehohlk*cenovf                                   3f6d
           + 2*cmapl);
                                                                        3f6d1
    %lock sets%                                                         3f7
        DECLARE dlsdef = (                                              3f7a
        $vlsfld, $lsr, $lsinit, 0, 0,                                   3f7b
```

```
    (cmlkidl+cmpidl)*crcwl + clsmnh*cmnhd + clsmxh*cmxhd,          3f7c
    egmhls*cennex + ehohls*cenovf                                  3f7d
       + 2*cmapl);
                                                                   3f7d1
%managers%                                                         3f8
    DECLARE dmgdef = (                                             3f8a
    $vmgfld, $mgr, $mginit, $mgterm, 0,                            3f8b
    cnoacs*crcwl + cmgmnh*cmnhd + cmgmxh*cmxhd,                    3f8c
    egmhmg*cennex + ehohmg*cenovf                                  3f8d
       + 2*cmapl);
                                                                   3f8d1
%packages%                                                         3f9
    DECLARE dpkdef = (                                             3f9a
    $vpkfld, $pkr, 0, 0, 0,                                        3f9b
    cpkmnh*cmnhd + cpkmxh*cmxhd,                                   3f9c
    egmhpk*cennex + ehohpk*cenovf                                  3f9d
       + 2*cmapl);
                                                                   3f9d1
%ports%                                                            3f10
    DECLARE dpodef = (                                             3f10a
    $vpofld, $por, $poinit, 0, 0,                                  3f10b
    4*crcwl + cpomnh*cmnhd + cpomxh*cmxhd,                         3f10c
    egmhpo*cennex + ehohpo*cenovf                                  3f10d
       + 2*cmapl);
                                                                   3f10d1
%processors%                                                       3f11
    DECLARE dprdef = (                                             3f11a
```

```
        $vprfld, $prr, $prinit, $prterm, $prhand,                    3f11b
        2*crcwl + cprmnh*cmnhd + cprmxh*cmxhd,                       3f11c
        egmhpr*cennex + ehohpr*cenovf                                3f11d
           + 2*cmapl);
                                                                     3f11d1
    %processes%                                                      3f12
        DECLARE dpsdef = (                                           3f12a
        $vpsfld, $psr, 0, $psterm, $pshand,                          3f12b
        cpsmnh*cmnhd + cpsmxh*cmxhd,                                 3f12c
        egmhps*cennex + ehohps*cenovf                                3f12d
           + 2*cmapl);
                                                                     3f12d1
    %segments%                                                       3f13
        DECLARE dsgdef = (                                           3f13a
        $vsgfld, $sgr, $cainit, $caterm, 0,                          3f13b
        csgmnh*cmnhd + csgmxh*cmxhd,                                 3f13c
        egmhsg*cennex + ehohsg*cenovf                                3f13d
           + 2*cmapl);
                                                                     3f13d1
    %subprocesses%                                                   3f14
        DECLARE dsudef = (                                           3f14a
        $vsufld, $sur, $suinit, $suterm, $suhand,                    3f14b
        (1+(cmgmxh-cmgmnh+1))*crcwl + csumnh*cmnhd + csumxh*cmxhd,   3f14c
        egmhsu*cennex + ehohsu*cenovf                                3f14d
           + 2*cmapl);
                                                                     3f14d1
    %system calls%                                                   3f15
```

```
        DECLARE dsydef = (                                          3f15a
        $vsyfld, $syr, 0, $syterm, 0,                               3f15b
        csymnh*cmnhd + csymxh*cmxhd,                                3f15c
        egmhsy*cennex + ehohsy*cenovf                               3f15d
           + 2*cmapl);
                                                                    3f15d1
     %user calls%                                                   3f16
        DECLARE dusdef = (                                          3f16a
        $vusfld, $usr, $usinit, $usterm, 0,                         3f16b
        cusmnh*cmnhd + cusmxh*cmxhd,                                3f16c
        egmhus*cennex + ehohus*cenovf                               3f16d
           + 2*cmapl);
                                                                    3f16d1
     %timers%                                                       3f17
        DECLARE dtmdef = (                                          3f17a
        $vtmfld, $tmr, 0, 0, 0,                                     3f17b
        ctmmnh*cmnhd + ctmmxh*cmxhd,                                3f17c
        egmhtm*cennex + ehohtm*cenovf                               3f17d
           + 2*cmapl);
                                                                    3f17d1
  %DPS operation definition table%                                  3g
     DECLARE EXTERNAL doprtbl = (                                   3g1
     $ivdps, $rrdps, $drdps, $gtdps, $ptdps, $pgdps, $erdps);
                                                                    3g2
  %system call definition table%                                    3h
     % model                                                       3h1
        addr, argcnt*cargs + rescnt*cress,                          3h1a
```

```
        arguments                                                       3h1b
           carg1*((ac-1)*cacn + loc*cpos + type) +                      3h1b1
           carg2*((ac-1)*cacn + loc*cpos + type) +                      3h1b2
           carg3*((ac-1)*cacn + loc*cpos + type),                       3h1b3
           carg4*((ac-1)*cacn + loc*cpos + type) +                      3h1b4
           carg5*((ac-1)*cacn + loc*cpos + type) +                      3h1b5
           carg6*((ac-1)*cacn + loc*cpos + type),                       3h1b6
        results                                                         3h1c
           cres1*((ac-1)*cacn + loc*cpos + type) +                      3h1c1
           cres2*((ac-1)*cacn + loc*cpos + type) +                      3h1c2
           cres3*((ac-1)*cacn + loc*cpos + type), %                     3h1c3

     % crtps  (1:charstr, RH2:usrinf, LH2:[struc], RH3:index,
              RH4:[list], LH4:[list], LH3:[index], :list
              => RH1:index, LH1: [list]) %                              3h2
        DECLARE EXTERNAL dsyctbl = (scrtps, 8*cargs + 2*cress,          3h2a
        %arguments%                                                     3h2b
           carg1*((1-1)*cacn + cwh*cpos + cucstr) +                     3h2b1
           carg2*((2-1)*cacn + crh*cpos + cusrinf) +                    3h2b2
           carg3*((2-1)*cacn + clh*cpos + cstruc+cempok),               3h2b3
           carg4*((3-1)*cacn + crh*cpos + cindex) +                     3h2b4
           carg5*((4-1)*cacn + crh*cpos + clistof+cucstr+cempok) +      3h2b5
           carg6*((4-1)*cacn + clh*cpos + clist+cempok),                3h2b6
           carg7*((3-1)*cacn + clh*cpos + cindex+cempok) +              3h2b7
           carg8*((1-1)*cacn + cno*cpos + cdumlist),                    3h2b8
        %results%                                                       3h2c
```

```
        cres1*((1-1)*cacn + crh*cpos + cindex) +                    3h2c1
        cres2*((1-1)*cacn + clh*cpos + clistof+cindex+cempok));
                                                                    3h2c2
    % delps  (1:[index] -> 1:integer) %                            3h3
      DECLARE ddelps = ($delps, 1*cargs + 1*cress,                  3h3a
      %argument%                                                    3h3b
        carg1*((1-1)*cacn + cwh*cpos + cindex+cempok), 0, 0,        3h3b1
      %result%                                                      3h3c
        cres1*((1-1)*cacn + cwh*cpos + cinteger));
                                                                    3h3c1
    %+PDP10% % itdps  (RH1:index, LH1:[struc], RH2:index,
    LH2:[struc],
            RH3:index, FG3:integer -> 1:integer, LH2:index,
            RH2:index) %                                            3h4
      DECLARE ditdps = ($itdps, 6*cargs + 3*cress,                  3h4a
      %arguments%                                                   3h4b
        carg1*((1-1)*cacn + crh*cpos + cindex) +                    3h4b1
        carg2*((1-1)*cacn + clh*cpos + cstruc+cempok) +             3h4b2
        carg3*((2-1)*cacn + crh*cpos + cindex),                     3h4b3
        carg4*((2-1)*cacn + clh*cpos + cstruc+cempok) +             3h4b4
        carg5*((3-1)*cacn + crh*cpos + cindex) +                    3h4b5
        carg6*((3-1)*cacn + cfg*cpos + cinteger), 0,                3h4b6
      %results%                                                     3h4c
        cres1*((1-1)*cacn + crh*cpos + cinteger) +                  3h4c1
        cres2*((2-1)*cacn + clh*cpos + cindex) +                    3h4c2
        cres3*((2-1)*cacn + crh*cpos + cindex));%+PDP10%
                                                                    3h4c3
    %+PDP10% % sepps  (1:[index] -> 1:integer, 2:integer) %         3h5
```

```
    DECLARE dsepps = ($sepps, 1*cargs + 2*cress,                    3h5a
    %arguments%                                                     3h5b
       carg1*((1-1)*cacn + cwh*cpos + cindex+cempok), 0, 0,         3h5b1
    %results%                                                       3h5c
       cres1*((1-1)*cacn + cwh*cpos + cinteger) +                   3h5c1
       cres2*((2-1)*cacn + cwh*cpos + cinteger));%+PDP10%           3h5c2

% opnpk  (RH1:index, RH2:chstlist, LH2:[list], LH1:index,
          :list -> 1:idxlist) %                                     3h6
    DECLARE dopnpk = ($opnpk, 5*cargs + 1*cress,                    3h6a
    %arguments%                                                     3h6b
       carg1*((1-1)*cacn + crh*cpos + cindex) +                     3h6b1
       carg2*((2-1)*cacn + crh*cpos + clistof+cucstr) +             3h6b2
       carg3*((2-1)*cacn + clh*cpos + clist+cempok),                3h6b3
       carg4*((1-1)*cacn + clh*cpos + cindex) +                     3h6b4
       carg5*((1-1)*cacn + cno*cpos + cdumlist), 0,                 3h6b5
    %results%                                                       3h6c
       cres1*((1-1)*cacn + cwh*cpos + clistof+cindex));             3h6c1

% clspk  (LH1:index, RH1:idxlist, :list
          => 1:intlist) %                                           3h7
    DECLARE dclspk = ($clspk, 3*cargs + 1*cress,                    3h7a
    %arguments%                                                     3h7b
       carg1*((1-1)*cacn + clh*cpos + cindex) +                     3h7b1
       carg2*((1-1)*cacn + crh*cpos + clistof+cindex) +             3h7b2
       carg3*((1-1)*cacn + cno*cpos + cdumlist), 0, 0,              3h7b3
    %results%                                                       3h7c
```

```
        cres1*((1-1)*cacn + cwh*cpos + clistof+cinteger));                3h7c1

    % calpe  (RH1:psel, RH2:[list], LH2:[struc],
             LH1:[struc], :list, 3:index => LH1:index,
             RH1:[list], 2:integer) %                                     3h8

      DECLARE dcalpe = (scalpe, 6*cargs + 3*cress,                        3h8a

      %arguments%                                                         3h8b

         carg1*((1-1)*cacn + crh*cpos + cintpsel) +                       3h8b1

         carg2*((3-1)*cacn + cwh*cpos + cindex) +                         3h8b2

         carg3*((2-1)*cacn + crh*cpos + clist+cempok),                    3h8b3

         carg4*((2-1)*cacn + clh*cpos + cstruc+cempok) +                  3h8b4

         carg5*((1-1)*cacn + clh*cpos + cstruc+cempok) +                  3h8b5

         carg6*((1-1)*cacn + cno*cpos + cdumlist), 0,                     3h8b6

      %results%                                                           3h8c

         cres1*((1-1)*cacn + clh*cpos + cindex) +                         3h8c1

         cres2*((1-1)*cacn + crh*cpos + clist+cempok) +                   3h8c2

         cres3*((2-1)*cacn + cwh*cpos + cinteger));                       3h8c3

    % vispe  (RH1:index, RH2:[list], LH2:[struc],
             LH1:[struc], :list => LH1:index,
             RH1:[list]) %                                                3h9

      DECLARE dvispe = (svispe, 5*cargs + 2*cress,                        3h9a

      %arguments%                                                         3h9b

         carg1*((1-1)*cacn + crh*cpos + cindex) +                         3h9b1

         carg2*((2-1)*cacn + crh*cpos + clist+cempok) +                   3h9b2

         carg3*((2-1)*cacn + clh*cpos + cstruc+cempok),                   3h9b3

         carg4*((1-1)*cacn + clh*cpos + cstruc+cempok) +                  3h9b4

         carg5*((1-1)*cacn + cno*cpos + cdumlist), 0,                     3h9b5
```

```
    %results%                                                          3h9c
       cres1*((1-1)*cacn + clh*cpos + cindex) +                        3h9c1
       cres2*((1-1)*cacn + crh*cpos + clist+cempok));
                                                                       3h9c2
  % aloch  (RH1:psel, LH1:index -> 1:index) %                         3h10
    DECLARE daloch = ($aloch, 2*cargs + 1*cress,                       3h10a
    %arguments%                                                        3h10b
       carg1*((1-1)*cacn + crh*cpos + cintpsel) +                      3h10b1
       carg2*((1-1)*cacn + clh*cpos + cindex), 0, 0,                   3h10b2
    %cindex%                                                           3h10c
       cres1*((1-1)*cacn + cwh*cpos + cindex));
                                                                       3h10c1
  % relch  (1:[index] -> 1:integer) %                                  3h11
    DECLARE drelch = ($relch, 1*cargs + 1*cress,                       3h11a
    %arguments%                                                        3h11b
       carg1*((1-1)*cacn + cwh*cpos + cindex+cempok), 0, 0,            3h11b1
    %results%                                                          3h11c
       cres1*((1-1)*cacn + cwh*cpos + cinteger));
                                                                       3h11c1
  % acqpe  (1:index, :list -> LH1:index, RH1:[list]) %                 3h12
    DECLARE dacqpe = ($acqpe, 2*cargs + 2*cress,                       3h12a
    %arguments%                                                        3h12b
       carg1*((1-1)*cacn + cwh*cpos + cindex) +                        3h12b1
       carg2*((1-1)*cacn + cno*cpos + cdumlist), 0, 0,                 3h12b2
    %results%                                                          3h12c
       cres1*((1-1)*cacn + clh*cpos + cindex) +                        3h12c1
```

```
        cres2*((1-1)*cacn + crh*cpos + clist+cempok));                3h12c2

    % relpe  (RH1:index, RH2:[list], LH2:[struc],                      3h13
              LH1:[struc], 3:index) %
      DECLARE drelpe = ($relpe, 5*cargs + 0*cress,                    3h13a
      %arguments%                                                     3h13b
        carg1*((1-1)*cacn + crh*cpos + cindex) +                      3h13b1
        carg2*((2-1)*cacn + crh*cpos + clist+cempok) +                3h13b2
        carg3*((2-1)*cacn + clh*cpos + cstruc+cempok),                3h13b3
        carg4*((1-1)*cacn + clh*cpos + cstruc+cempok) +               3h13b4
        carg5*((3-1)*cacn + cwh*cpos + cindex), 0,                    3h13b5
      %results%                                                       3h13c
        0);                                                           3h13c1

    %+PDP10% % intpe  (1:[index]) %                                   3h14
      DECLARE dintpe = ($intpe, 1*cargs + 0*cress,                    3h14a
      %arguments%                                                     3h14b
        carg1*((1-1)*cacn + cwh*cpos + cindex+cempok), 0, 0,          3h14b1
      %results%                                                       3h14c
        0);%+PDP10%                                                   3h14c1

    %+PDP10% % rsmpe  (1:[index]) %                                   3h15
      DECLARE drsmpe = ($rsmpe, 1*cargs + 0*cress,                    3h15a
      %arguments%                                                     3h15b
        carg1*((1-1)*cacn + cwh*cpos + cindex+cempok), 0, 0,          3h15b1
      %results%                                                       3h15c
```

```
        0);%+PDP10%                                                    3h15c1
    %+PDP10% % ntepe  (RH1:index, LH1:[struc]) %                       3h16
       DECLARE dntepe = ($ntepe, 2*cargs + 0*cress,                    3h16a
       %arguments%                                                     3h16b
          carg1*((1-1)*cacn + crh*cpos + cindex) +                     3h16b1
          carg2*((1-1)*cacn + clh*cpos + cstruc+cempok), 0, 0,         3h16b2
       %results%                                                       3h16c
          0);                                                          3h16c1
    % hlppe  (RH1:index, LH1:[struc] => 1:[struc]) %                   3h17
       DECLARE dhlppe = ($hlppe, 2*cargs + 1*cress,                    3h17a
       %arguments%                                                     3h17b
          carg1*((1-1)*cacn + crh*cpos + cindex) +                     3h17b1
          carg2*((1-1)*cacn + clh*cpos + cstruc+cempok), 0, 0,         3h17b2
       %results%                                                       3h17c
          cres1*((1-1)*cacn + cwh*cpos + calostruc+cempok));           3h17c1
    %+PDP10% % crtdt  (RH1:dsel, 2:[struc], LH1:index) %               3h18
       DECLARE dcrtdt = ($crtdt, 3*cargs + 0*cress,                    3h18a
       %arguments%                                                     3h18b
          carg1*((1-1)*cacn + crh*cpos + cdsel) +                      3h18b1
          carg2*((2-1)*cacn + cwh*cpos + cstruc+cempok) +              3h18b2
          carg3*((1-1)*cacn + clh*cpos + cindex), 0, 0,                3h18b3
       %results%                                                       3h18c
          0);%+PDP10%                                                  3h18c1
```

```
%+PDP10% % deldt  (1:dsel) %                                        3h19
   DECLARE ddeldt = ($deldt, 1*cargs + 0*cress,                     3h19a
   %arguments%                                                      3h19b
      carg1*((1-1)*cacn + cwh*cpos + cdsel), 0, 0,                  3h19b1
   %results%                                                        3h19c
      0);%+PDP10%
                                                                    3h19c1
%+PDP10% % rddt   (1:dsel -> 1:[struc]) %                           3h20
   DECLARE drddt = ($rddt, 1*cargs + 1*cress,                       3h20a
   %arguments%                                                      3h20b
      carg1*((1-1)*cacn + cwh*cpos + cdsel), 0, 0,                  3h20b1
   %results%                                                        3h20c
      cres1*((1-1)*cacn + cwh*cpos +
      calostruc+cempok));%+PDP10%
                                                                    3h20c1
%+PDP10% % wrdt   (RH1:dsel, LH1:[struc]) %                         3h21
   DECLARE dwrdt = ($wrdt, 2*cargs + 0*cress,                       3h21a
   %arguments%                                                      3h21b
      carg1*((1-1)*cacn + crh*cpos + cdsel) +                       3h21b1
      carg2*((1-1)*cacn + clh*cpos + cstruc+cempok), 0, 0,          3h21b2
   %results%                                                        3h21c
      0);%+PDP10%
                                                                    3h21c1
%+PDP10% % lckdt  (RH1:dsel, RH2:index, LH1:index, FG2:integer
        -> 1:index) %                                               3h22
   DECLARE dlckdt = ($lckdt, 4*cargs + 1*cress,                     3h22a
   %arguments%                                                      3h22b
```

```
        carg1*((1-1)*cacn + crh*cpos + cdsel) +                    3h22b1
        carg2*((2-1)*cacn + crh*cpos + cindex) +                   3h22b2
        carg3*((1-1)*cacn + clh*cpos + cindex),                    3h22b3
        carg4*((2-1)*cacn + cfg*cpos + cinteger), 0,               3h22b4
     %results%                                                     3h22c
        cres1*((1-1)*cacn + cwh*cpos + cindex));%+PDP10%
                                                                   3h22c1
  %+PDP10% % ulkdt  (RH1:dsel, LH1:index) %                        3h23
     DECLARE dulkdt = ($ulkdt, 2*cargs + 0*cress,                  3h23a
     %arguments%                                                   3h23b
        carg1*((1-1)*cacn + crh*cpos + cdsel) +                    3h23b1
        carg2*((1-1)*cacn + clh*cpos + cindex), 0, 0,              3h23b2
     %results%                                                     3h23c
        0);%+PDP10%
                                                                   3h23c1
  %+PDP10% % crtch  (LH1:index, :empty, RH1:index, :empty,
  2:index
           => 1:index, LH2:index, RH2:index) %                     3h24
     DECLARE dcrtch = ($crtch, 5*cargs + 3*cress,                  3h24a
     %arguments%                                                   3h24b
        carg1*((1-1)*cacn + clh*cpos + cindex) +                   3h24b1
        carg2*((1-1)*cacn + cno*cpos + cdumemp) +                  3h24b2
        carg3*((1-1)*cacn + crh*cpos + cindex),                    3h24b3
        carg4*((1-1)*cacn + cno*cpos + cdumemp) +                  3h24b4
        carg5*((2-1)*cacn + cwh*cpos + cindex), 0,                 3h24b5
     %results%                                                     3h24c
        cres1*((1-1)*cacn + cwh*cpos + cindex) +                   3h24c1
```

```
        cres2*((2-1)*cacn + clh*cpos + cindex) +                    3h24c2
        cres3*((2-1)*cacn + crh*cpos + cindex));%+PDP10%
                                                                    3h24c3
    %+PDP10% % delch  (1:[index]) %                                 3h25
      DECLARE ddelch = ($delch, 1*cargs + 0*cress,                  3h25a
      %arguments%                                                   3h25b
        carg1*((1-1)*cacn + cwh*cpos + cindex+cempok), 0, 0,        3h25b1
      %results%                                                     3h25c
        0);%+PDP10%
                                                                    3h25c1
    % crtsp  (1:charstr, RH2:[struc], LH2:index, 3:index,
              :empty -> 1:index) %                                  3h26
      DECLARE dcrtsp = ($crtsp, 5*cargs + 1*cress,                  3h26a
      %arguments%                                                   3h26b
        carg1*((1-1)*cacn + cwh*cpos + cucstr) +                    3h26b1
        carg2*((2-1)*cacn + crh*cpos + cstruc+cempok) +             3h26b2
        carg3*((2-1)*cacn + clh*cpos + cindex),                     3h26b3
        carg4*((3-1)*cacn + cwh*cpos + cindex) +                    3h26b4
        carg5*((1-1)*cacn + cno*cpos + cdumemp), 0,                 3h26b5
      %results%                                                     3h26c
        cres1*((1-1)*cacn + cwh*cpos + cindex));
                                                                    3h26c1
    % delsp  (1:[index], :empty -> 1:integer) %                     3h27
      DECLARE ddelsp = ($delsp, 2*cargs + 1*cress,                  3h27a
      %arguments%                                                   3h27b
        carg1*((1-1)*cacn + cwh*cpos + cindex+cempok) +             3h27b1
        carg2*((1-1)*cacn + cno*cpos + cdumemp), 0, 0,              3h27b2
```

```
    %results%                                                    3h27c
       cres1*((1-1)*cacn + cwh*cpos + cinteger));
                                                                 3h27c1
% crtpr  (RH1:index, LH2:[struc], LH1:index, RH2:index,
          :empty => 1:index) %                                   3h28
    DECLARE dcrtpr = ($crtpr, 5*cargs + 1*cress,                 3h28a
    %arguments%                                                  3h28b
       carg1*((1-1)*cacn + crh*cpos + cindex) +                  3h28b1
       carg2*((2-1)*cacn + clh*cpos + cstruc+cempok) +           3h28b2
       carg3*((1-1)*cacn + clh*cpos + cindex),                   3h28b3
       carg4*((2-1)*cacn + crh*cpos + cindex) +                  3h28b4
       carg5*((1-1)*cacn + cno*cpos + cdumemp), 0,               3h28b5
    %results%                                                    3h28c
       cres1*((1-1)*cacn + cwh*cpos + cindex));
                                                                 3h28c1
% delpr  (1:[index], :empty => 1:integer) %                      3h29
    DECLARE ddelpr = ($delpr, 2*cargs + 1*cress,                 3h29a
    %arguments%                                                  3h29b
       carg1*((1-1)*cacn + cwh*cpos + cindex+cempok) +           3h29b1
       carg2*((1-1)*cacn + cno*cpos + cdumemp), 0, 0,            3h29b2
    %results%                                                    3h29c
       cres1*((1-1)*cacn + cwh*cpos + cinteger));
                                                                 3h29c1
% sipr   (1:charstr, RH2:[chstlist], LH3:integer, RH3:integer,
          FG2:integer => LH1:[struc], RH1:index,
          FG2:integer) %                                         3h30
    DECLARE dsipr = ($sipr, 5*cargs + 3*cress,                   3h30a
    %arguments%                                                  3h30b
```

```
        carg1*((1-1)*cacn + cwh*cpos + cucstr) +                    3h30b1
        carg2*((2-1)*cacn + crh*cpos + clistof+cucstr+cempok) +     3h30b2
        carg3*((3-1)*cacn + clh*cpos + cinteger),                   3h30b3
        carg4*((3-1)*cacn + crh*cpos + cinteger) +                  3h30b4
        carg5*((2-1)*cacn + cfg*cpos + cinteger), 0,                3h30b5
    %results%                                                       3h30c
        cres1*((1-1)*cacn + clh*cpos + calostruc+cempok) +          3h30c1
        cres2*((1-1)*cacn + crh*cpos + cindex+cempok) +             3h30c2
        cres3*((2-1)*cacn + cfg*cpos + cinteger));
                                                                    3h30c3
% rdypr  () %                                                      3h31
    DECLARE drdypr = ($rdypr, 0*cargs + 0*cress,                    3h31a
    %arguments%                                                     3h31b
        0, 0, 0,                                                    3h31b1
    %results%                                                       3h31c
        0);
                                                                    3h31c1
%+PDP10% % sndch  (RH1:index, LH1:data) %                          3h32
    DECLARE dsndch = ($sndch, 2*cargs + 0*cress,                    3h32a
    %arguments%                                                     3h32b
        carg1*((1-1)*cacn + crh*cpos + cindex) +                    3h32b1
        carg2*((1-1)*cacn + clh*cpos + cdata), 0, 0,                3h32b2
    %results%                                                       3h32c
        0);%+PDP10%
                                                                    3h32c1
%+PDP10% % rcvch  (1:index -> 1:data) %                            3h33
```

```
    DECLARE drcvch = ($rcvch, 1*cargs + 1*cress,                    3h33a
    %arguments%                                                     3h33b
       carg1*((1-1)*cacn + cwh*cpos + cindex), 0, 0,                3h33b1
    %results%                                                       3h33c
       cres1*((1-1)*cacn + cwh*cpos + calodata));%+PDP10%           3h33c1
%+PDP10% % crtlk  (1:index -> 1:index) %                            3h34
    DECLARE dcrtlk = ($crtlk, 1*cargs + 1*cress,                    3h34a
    %arguments%                                                     3h34b
       carg1*((1-1)*cacn + cwh*cpos + cindex), 0, 0,                3h34b1
    %results%                                                       3h34c
       cres1*((1-1)*cacn + cwh*cpos + cindex));%+PDP10%             3h34c1
%+PDP10% % dellk  (1:[index]) %                                     3h35
    DECLARE ddellk = ($dellk, 1*cargs + 0*cress,                    3h35a
    %arguments%                                                     3h35b
       carg1*((1-1)*cacn + cwh*cpos + cindex+cempok), 0, 0,         3h35b1
    %results%                                                       3h35c
       0);%+PDP10%                                                  3h35c1
%+PDP10% % setlk  (RH1:index, RH2:index, LH1:index, FG2:integer
          -> 1:index) %                                             3h36
    DECLARE dsetlk = ($setlk, 4*cargs + 1*cress,                    3h36a
    %arguments%                                                     3h36b
       carg1*((1-1)*cacn + crh*cpos + cindex) +                     3h36b1
       carg2*((2-1)*cacn + crh*cpos + cindex) +                     3h36b2
       carg3*((1-1)*cacn + clh*cpos + cindex),                      3h36b3
```

```
        carg4*((2-1)*cacn + cfg*cpos + cinteger), 0,                    3h36b4
    %results%                                                           3h36c
        cres1*((1-1)*cacn + cwh*cpos + cindex));%+PDP10%
                                                                        3h36c1
%+PDP10% % remlk  (LH1:index, RH1:index) %                             3h37
    DECLARE dremlk = ($remlk, 2*cargs + 0*cress,                        3h37a
    %arguments%                                                         3h37b
        carg1*((1-1)*cacn + clh*cpos + cindex) +                        3h37b1
        carg2*((1-1)*cacn + crh*cpos + cindex), 0, 0,                   3h37b2
    %results%                                                           3h37c
        0);%+PDP10%
                                                                        3h37c1
%+PDP10% % crtev  (LH1:index, 2:index, RH1:integer => 1:index)
%                                                                       3h38
    DECLARE dcrtev = ($crtev, 3*cargs + 1*cress,                        3h38a
    %arguments%                                                         3h38b
        carg1*((1-1)*cacn + clh*cpos + cindex) +                        3h38b1
        carg2*((2-1)*cacn + cwh*cpos + cindex) +                        3h38b2
        carg3*((1-1)*cacn + crh*cpos + cinteger), 0, 0,                 3h38b3
    %results%                                                           3h38c
        cres1*((1-1)*cacn + cwh*cpos + cindex));%+PDP10%
                                                                        3h38c1
%+PDP10% % delev  (1:[index]) %                                        3h39
    DECLARE ddelev = ($delev, 1*cargs + 0*cress,                        3h39a
    %arguments%                                                         3h39b
        carg1*((1-1)*cacn + cwh*cpos + cindex), 0, 0,                   3h39b1
    %results%                                                           3h39c
```

```
        0);%+PDP10%                                                    3h39c1
    %+PDP10% % sigev  (1:index, 2:integer, :empty) %                   3h40
       DECLARE dsigev = ($sigev, 3*cargs + 0*cress,                    3h40a
       %arguments%                                                     3h40b
          carg1*((1-1)*cacn + cwh*cpos + cindex) +                     3h40b1
          carg2*((2-1)*cacn + cwh*cpos + cinteger) +                   3h40b2
          carg3*((1-1)*cacn + cno*cpos + cdumemp), 0, 0,               3h40b3
       %results%                                                       3h40c
          0);%+PDP10%                                                  3h40c1
    %+PDP10% % tstev  (1:index -> 1:integer, 2:integer) %              3h41
       DECLARE dtstev = ($tstev, 1*cargs + 2*cress,                    3h41a
       %arguments%                                                     3h41b
          carg1*((1-1)*cacn + cwh*cpos + cindex), 0, 0,                3h41b1
       %results%                                                       3h41c
          cres1*((1-1)*cacn + cwh*cpos + cinteger) +                   3h41c1
          cres2*((2-1)*cacn + cwh*cpos + cinteger));%+PDP10%           3h41c2
    %+PDP10% % waiev  (1:idxlist -> 2:integer, LH1:index,
    RH1:integer) %                                                     3h42
       DECLARE dwaiev = ($waiev, 1*cargs + 3*cress,                    3h42a
       %arguments%                                                     3h42b
          carg1*((1-1)*cacn + cwh*cpos + clistof+cindex), 0, 0,        3h42b1
       %results%                                                       3h42c
          cres1*((2-1)*cacn + cwh*cpos + cinteger) +                   3h42c1
          cres2*((1-1)*cacn + clh*cpos + cindex) +                     3h42c2
```

```
        cres3*((1-1)*cacn + crh*cpos + cinteger));%+PDP10%                3h42c3
    %+PDP10% % infps  (1:index -> 1:integer) %                            3h43
      DECLARE dinfps = ($infps, 1*cargs + 1*cress,                        3h43a
      %arguments%                                                         3h43b
        carg1*((1-1)*cacn + cwh*cpos + cindex), 0, 0,                     3h43b1
      %results%                                                           3h43c
        cres1*((1-1)*cacn + cwh*cpos + cinteger));%+PDP10%                3h43c1
    %+PDP10% % sopr   () %                                                3h44
      DECLARE dsopr = ($sopr, 0*cargs + 0*cress,                          3h44a
      %arguments%                                                         3h44b
        0, 0, 0,                                                          3h44b1
      %results%                                                           3h44c
        0);%+PDP10%                                                       3h44c1
    %+PDP10% % itdfk  (RH1:integer, LH1:[struc], :empty -> 1:index)
    %                                                                     3h45
      DECLARE ditdfk = ($itdfk, 3*cargs + 1*cress,                        3h45a
      %arguments%                                                         3h45b
        carg1*((1-1)*cacn + crh*cpos + cinteger) +                        3h45b1
        carg2*((1-1)*cacn + clh*cpos + cstruc+cempok) +                   3h45b2
        carg3*((1-1)*cacn + cno*cpos + cdumemp), 0, 0,                    3h45b3
      %results%                                                           3h45c
        cres1*((1-1)*cacn + cwh*cpos + cindex));%+PDP10%                  3h45c1
    %+PDP10% % sepfk  (1:index, :empty) %                                 3h46
```

```
    DECLARE dsepfk = ($sepfk, 2*cargs + 0*cress,                    3h46a
    %arguments%                                                     3h46b
       carg1*((1-1)*cacn + cwh*cpos + cindex) +                     3h46b1
       carg2*((1-1)*cacn + cno*cpos + cdumemp), 0, 0,               3h46b2
    %results%                                                       3h46c
       0);%+PDP10%
                                                                    3h46c1
% sigpe  (1:index, RH2:[list], LH2:[struc]) %                       3h47
    DECLARE dsigpe = ($sigpe, 3*cargs + 0*cress,                    3h47a
    %arguments%                                                     3h47b
       carg1*((1-1)*cacn + cwh*cpos + cindex) +                     3h47b1
       carg2*((2-1)*cacn + crh*cpos + clist+cempok) +               3h47b2
       carg3*((2-1)*cacn + clh*cpos + cstruc+cempok), 0, 0,         3h47b3
    %results%                                                       3h47c
       0);
                                                                    3h47c1
% setmr  (1:integer, :empty, RH2:index, LH2:index => 1:index) %     3h48
    DECLARE dsetmr = ($setmr, 4*cargs + 1*cress,                    3h48a
    %arguments%                                                     3h48b
       carg1*((1-1)*cacn + cwh*cpos + cinteger) +                   3h48b1
       carg2*((1-1)*cacn + cno*cpos + cdumemp) +                    3h48b2
       carg3*((2-1)*cacn + crh*cpos + cindex),                      3h48b3
       carg4*((2-1)*cacn + clh*cpos + cindex), 0,                   3h48b4
    %results%                                                       3h48c
       cres1*((1-1)*cacn + cwh*cpos + cindex));
                                                                    3h48c1
```

```
    % tstmr  (FG1:integer, RH1:index => 1:integer, 2:integer) %          3h49
      DECLARE dtstmr = ($tstmr, 2*cargs + 2*cress,                       3h49a
      %arguments%                                                        3h49b
        carg1*((1-1)*cacn + cfg*cpos + cinteger) +                       3h49b1
        carg2*((1-1)*cacn + crh*cpos + cindex), 0, 0,                    3h49b2
      %results%                                                          3h49c
        cres1*((1-1)*cacn + cwh*cpos + cinteger) +                       3h49c1
        cres2*((2-1)*cacn + cwh*cpos + cinteger));
                                                                         3h49c2
    % setrc  (1:index, 2:integer) %                                     3h50
      DECLARE dsetrc = (ssetrc, 2*cargs + 0*cress,                       3h50a
      %arguments%                                                        3h50b
        carg1*((1-1)*cacn + cwh*cpos + cindex) +                         3h50b1
        carg2*((2-1)*cacn + cwh*cpos + cinteger), 0, 0,                  3h50b2
      %results%                                                          3h50c
        0);
                                                                         3h50c1
  %user call definition table%                                           3i
    % model                                                            3i1
      priority, argcnt*cargs + rescnt*cress,                             3i1a
      arguments                                                          3i1b
        carg1*((ac-1)*cacn + loc*cpos + type) +                          3i1b1
        carg2*((ac-1)*cacn + loc*cpos + type) +                          3i1b2
        carg3*((ac-1)*cacn + loc*cpos + type),                           3i1b3
        carg4*((ac-1)*cacn + loc*cpos + type) +                          3i1b4
        carg5*((ac-1)*cacn + loc*cpos + type) +                          3i1b5
```

```
        carg6*((ac-1)*cacn + loc*cpos + type),                          3i1b6
    results                                                             3i1c
        cres1*((ac-1)*cacn + loc*cpos + type) +                         3i1c1
        cres2*((ac-1)*cacn + loc*cpos + type) +                         3i1c2
        cres3*((ac-1)*cacn + loc*cpos + type), %                        3i1c3
% prso   () %                                                          3i2
    DECLARE EXTERNAL dusctbl = (cjuprio, 0*cargs + 0*cress,             3i2a
    %arguments%                                                         3i2b
        0, 0,                                                           3i2b1
    %results%                                                           3i2c
        0);                                                             3i2c1
% inipk  (1:index -> 1:index) %                                        3i3
    DECLARE dinipk = (cjuprio, 1*cargs + 1*cress,                       3i3a
    %arguments%                                                         3i3b
        carg1*((1-1)*cacn + cwh*cpos + cindex), 0,                      3i3b1
    %results%                                                           3i3c
        cres1*((1-1)*cacn + cwh*cpos + cindex));                        3i3c1
% trmpk  (1:index) %                                                   3i4
    DECLARE dtrmpk = (cjuprio, 1*cargs + 0*cress,                       3i4a
    %arguments%                                                         3i4b
        carg1*((1-1)*cacn + cwh*cpos + cindex), 0,                      3i4b1
    %results%                                                           3i4c
        0);                                                             3i4c1
```

```
% pecal  (LH3:index, 1:index, 2:charstr, RH3:[list]
         -> LH1: index, RH1: [list]) %                                  3i5
   DECLARE dpecal = (cjuprio, 4*cargs + 2*cress,                        3i5a
   %arguments%                                                          3i5b
      carg1*((3-1)*cacn + clh*cpos + cindex) +                          3i5b1
      carg2*((1-1)*cacn + cwh*cpos + cindex) +                          3i5b2
      carg3*((2-1)*cacn + cwh*cpos + ccharstr),                         3i5b3
      carg4*((3-1)*cacn + crh*cpos + clist+cempok),                     3i5b4
   %results%                                                            3i5c
      cres1*((1-1)*cacn + cfg*cpos + cinteger) +                        3i5c1
      cres2*((1-1)*cacn + crh*cpos + clist+cempok));
                                                                        3i5c2
%+PDP10% % peint  (1:index) %                                           3i6
   DECLARE dpeint = (cjuprio, 1*cargs + 0*cress,                        3i6a
   %arguments%                                                          3i6b
      carg1*((1-1)*cacn + cwh*cpos + cindex), 0,                        3i6b1
   %results%                                                            3i6c
      0);%+PDP10%
                                                                        3i6c1
%+PDP10% % persm  (1:index) %                                           3i7
   DECLARE dpersm = (cjuprio, 1*cargs + 0*cress,                        3i7a
   %arguments%                                                          3i7b
      carg1*((1-1)*cacn + cwh*cpos + cindex), 0,                        3i7b1
   %results%                                                            3i7c
      0);%+PDP10%
                                                                        3i7c1
%+PDP10% % peabr  (1:index) %                                           3i8
```

```
    DECLARE dpeabr = (cjuprio, 1*cargs + 0*cress,                    318a
    %arguments%                                                      318b
       carg1*((1-1)*cacn + cwh*cpos + cindex), 0,                    318b1
    %results%                                                        318c
       0);%+PDP10%                                                   318c1
% pente  (LH1:index, RH1:index, RH2:index, LH2:[struc]) %            319
    DECLARE dpente = (cjuprio, 4*cargs + 0*cress,                    319a
    %arguments%                                                      319b
       carg1*((1-1)*cacn + clh*cpos + cindex) +                      319b1
       carg2*((1-1)*cacn + crh*cpos + cindex) +                      319b2
       carg3*((2-1)*cacn + crh*cpos + cindex),                       319b3
       carg4*((2-1)*cacn + clh*cpos + cstruc+cempok),                319b4
    %results%                                                        319c
       0);                                                           319c1
% pehlp  (LH1:index, RH1:index, RH2:index, LH2:[struc]
          => 1:[struc]) %                                            3110
    DECLARE dpehlp = (cjuprio, 4*cargs + 1*cress,                    3110a
    %arguments%                                                      3110b
       carg1*((1-1)*cacn + clh*cpos + cindex) +                      3110b1
       carg2*((1-1)*cacn + crh*cpos + cindex) +                      3110b2
       carg3*((2-1)*cacn + crh*cpos + cindex),                       3110b3
       carg4*((2-1)*cacn + clh*cpos + cstruc+cempok),                3110b4
    %results%                                                        3110c
       cres1*((1-1)*cacn + cwh*cpos + cstruc+cempok));               3110c1
```

```
%+PDP10% % lvrdt  (1:index, 2:charstr) %                              3i11
   DECLARE dlvrdt = (cjuprio, 2*cargs + 0*cress,                      3i11a
   %arguments%                                                        3i11b
      carg1*((1-1)*cacn + cwh*cpos + cindex) +                        3i11b1
      carg2*((2-1)*cacn + cwh*cpos + ccharstr), 0,                    3i11b2
   %results%                                                          3i11c
      0);%+PDP10%
                                                                      3i11c1
%+PDP10% % lrddt  (RH1:index, 2:charstr, LH1:[struc] =>
1:[struc]) %                                                          3i12
   DECLARE dlrddt = (cjuprio, 3*cargs + 1*cress,                      3i12a
   %arguments%                                                        3i12b
      carg1*((1-1)*cacn + crh*cpos + cindex) +                        3i12b1
      carg2*((2-1)*cacn + cwh*cpos + ccharstr) +                      3i12b2
      carg3*((1-1)*cacn + clh*cpos + cstruc+cempok), 0,               3i12b3
   %results%                                                          3i12c
      cres1*((1-1)*cacn + cwh*cpos + cstruc+cempok));%+PDP10%
                                                                      3i12c1
%+PDP10% % lwrdt  (RH1:index, 2:charstr, LH1:[struc],
3:[struc]) %                                                          3i13
   DECLARE dlwrdt = (cjuprio, 4*cargs + 0*cress,                      3i13a
   %arguments%                                                        3i13b
      carg1*((1-1)*cacn + crh*cpos + cindex) +                        3i13b1
      carg2*((2-1)*cacn + cwh*cpos + ccharstr) +                      3i13b2
      carg3*((1-1)*cacn + clh*cpos + cstruc+cempok),                  3i13b3
      carg4*((3-1)*cacn + cwh*cpos + cstruc+cempok),                  3i13b4
   %results%                                                          3i13c
```

```
        0);%+PDP10%                                                    3i13c1

    % okips  (RH1:index, LH1:[struc]) %                               3i14
      DECLARE dokips = (cjuprio, 2*cargs + 0*cress,                    3i14a
      %arguments%                                                      3i14b
        carg1*((1-1)*cacn + crh*cpos + cindex) +                       3i14b1
        carg2*((1-1)*cacn + clh*cpos + cstruc+cempok), 0,              3i14b2
      %results%                                                        3i14c
        0);
                                                                       3i14c1
    % oksps  (1:index) %                                              3i15
      DECLARE doksps = (cjuprio, 1*cargs + 0*cress,                    3i15a
      %arguments%                                                      3i15b
        carg1*((1-1)*cacn + cwh*cpos + cindex), 0,                     3i15b1
      %results%                                                        3i15c
        0);
                                                                       3i15c1
    % okopk  (RH1:index, 2:charstr, LH1:index, RH3:[struc],
    LH3:[index]) %                                                     3i16
      DECLARE dokopk = (cjuprio, 5*cargs + 1*cress,                    3i16a
      %arguments%                                                      3i16b
        carg1*((1-1)*cacn + crh*cpos + cindex) +                       3i16b1
        carg2*((2-1)*cacn + cwh*cpos + ccharstr) +                     3i16b2
        carg3*((1-1)*cacn + clh*cpos + cindex),                        3i16b3
        carg4*((3-1)*cacn + crh*cpos + cstruc+cempok) +                3i16b4
        carg5*((3-1)*cacn + clh*cpos + cindex+cempok),                 3i16b5
      %results%                                                        3i16c
```

```
        cres1*((1-1)*cacn + cwh*cpos + cinteger));                    3116c1
    % okcpk  (RH1:index, 2:charstr) %                                 3117
      DECLARE dokcpk = (cjuprio, 3*cargs + 0*cress,                   3117a
      %arguments%                                                     3117b
        carg1*((1-1)*cacn + crh*cpos + cindex) +                      3117b1
        carg2*((2-1)*cacn + cwh*cpos + ccharstr) +                    3117b2
        carg3*((1-1)*cacn + clh*cpos + cindex+cempok), 0,             3117b3
      %results%                                                       3117c
        0);                                                           3117c1
    % okcch  (1:index) %                                             3118
      DECLARE dokcch = (cjuprio, 1*cargs + 0*cress,                   3118a
      %arguments%                                                     3118b
        carg1*((1-1)*cacn + cwh*cpos + cindex), 0,                    3118b1
      %results%                                                       3118c
        0);                                                           3118c1
    % okdch  (1:index) %                                             3119
      DECLARE dokdch = (cjuprio, 1*cargs + 0*cress,                   3119a
      %arguments%                                                     3119b
        carg1*((1-1)*cacn + cwh*cpos + cindex), 0,                    3119b1
      %results%                                                       3119c
        0);                                                           3119c1
    % ntlch  (RH1:index, FG1:integer) %                              3120
      DECLARE dntlch = (cjuprio, 2*cargs + 0*cress,                   3120a
```

```
        %arguments%                                                    3i20b
           carg1*((1-1)*cacn + crh*cpos + cindex) +                    3i20b1
           carg2*((1-1)*cacn + cfg*cpos + cinteger), 0,                3i20b2
        %results%                                                      3i20c
           0);                                                         3i20c1
     % oklps  (RH1:index, 2:charstr, LH1:[struc]) %                    3i21
        DECLARE doklps = (cjuprio, 3*cargs + 0*cress,                  3i21a
        %arguments%                                                    3i21b
           carg1*((1-1)*cacn + crh*cpos + cindex) +                    3i21b1
           carg2*((2-1)*cacn + cwh*cpos + ccharstr) +                  3i21b2
           carg3*((1-1)*cacn + clh*cpos + cstruc+cempok), 0,           3i21b3
        %results%                                                      3i21c
           0);                                                         3i21c1
     % okups  (1:index) %                                             3i22
        DECLARE dokups = (cjuprio, 1*cargs + 0*cress,                  3i22a
        %arguments%                                                    3i22b
           carg1*((1-1)*cacn + cwh*cpos + cindex), 0,                  3i22b1
        %results%                                                      3i22c
           0);                                                         3i22c1
  %message definition table%                                           3j
     % model                                                          3j1
        addr, prmcnt*cprms,                                            3j1a
        header                                                         3j1b
```

```
        cprm1*(cindex) +                                        3j1b1
        cprm2*(cindex) +                                        3j1b2
    arguments                                                   3j1c
        cprm3*(type),                                           3j1c1
        cprm4*(type) +                                          3j1c2
        cprm5*(type) +                                          3j1c3
        cprm6*(type),                                           3j1c4
        cprm7*(type) +                                          3j1c5
        cprm8*(type) +                                          3j1c6
        cprm9*(type),                                           3j1c7
        cprm10*(type) +                                         3j1c8
        cprm11*(type) +                                         3j1c9
        cprm12*(type), %                                        3j1c10
% calpe  (index, [index], struc, [list], [list],
          [list], index) %                                      3j2
    DECLARE EXTERNAL dmsgtbl = (sxcalpe, 9*cprms,               3j2a
    %header%                                                    3j2b
        cprm1*(cindex) +                                        3j2b1
        cprm2*(cindex) +                                        3j2b2
    %arguments%                                                 3j2c
        cprm3*(cindex),                                         3j2c1
        cprm4*(cindex+cempok) +                                 3j2c2
        cprm5*(cstruc) +                                        3j2c3
        cprm6*(clist+cempok),                                   3j2c4
        cprm7*(clist+cempok) +                                  3j2c5
```

```
        cprm8*(clist+cempok) +                                  3j2c6
        cprm9*(cindex), 0);                                     3j2c7
    % recpe  (index, [list], [list], [list]) %                  3j3
      DECLARE dxrecpe = (sxrecpe, 6*cprms,                      3j3a
      %header%                                                  3j3b
        cprm1*(cindex) +                                        3j3b1
        cprm2*(cindex) +                                        3j3b2
      %arguments%                                               3j3c
        cprm3*(cindex),                                         3j3c1
        cprm4*(clist+cempok) +                                  3j3c2
        cprm5*(clist+cempok) +                                  3j3c3
        cprm6*(clist+cempok), 0, 0);                            3j3c4
    % rtnpe  (index, [list], [list], [list], index,
              integer) %                                        3j4
      DECLARE dxrtnpe = (sxrtnpe, 8*cprms,                      3j4a
      %header%                                                  3j4b
        cprm1*(cindex) +                                        3j4b1
        cprm2*(cindex) +                                        3j4b2
      %arguments%                                               3j4c
        cprm3*(cindex),                                         3j4c1
        cprm4*(clist+cempok) +                                  3j4c2
        cprm5*(clist+cempok) +                                  3j4c3
        cprm6*(clist+cempok),                                   3j4c4
        cprm7*(cindex) +                                        3j4c5
```

```
            cprm8*(cinteger), 0);                                   3j4c6

      % error  (index, ccharstr) %                                  3j5
         DECLARE dxerror = ($xerror, 4*cprms,                       3j5a
         %header%                                                   3j5b
            cprm1*(cindex) +                                        3j5b1
            cprm2*(cindex) +                                        3j5b2
         %arguments%                                                3j5c
            cprm3*(cindex),                                         3j5c1
            cprm4*(ccharstr), 0, 0);                                3j5c2
   %system procedure definition table%                              3k
      % model                                                       3k1
         addr, argcnt*cargs + rescnt*cress,                         3k1a
         arguments                                                  3k1b
            carg1*(type) +                                          3k1b1
            carg2*(type) +                                          3k1b2
            carg3*(type),                                           3k1b3
            carg4*(type) +                                          3k1b4
            carg5*(type) +                                          3k1b5
            carg6*(type),                                           3k1b6
         results                                                    3k1c
            cres1*(type) +                                          3k1c1
            cres2*(type) +                                          3k1c2
            cres3*(type), %                                         3k1c3
```

```
% inips  (charstr, usrinf, [struc], struc, [pkinfo], :list
          => struc, [list]) %                                          3k2
   DECLARE EXTERNAL dsyptbl = ($xinips, 6*cargs + 2*cress,             3k2a
   %arguments%                                                         3k2b
      carg1*(ccharstr) +                                               3k2b1
      carg2*(cusrinf) +                                                3k2b2
      carg3*(cstruc+cempok),                                           3k2b3
      carg4*(cstruc) +                                                 3k2b4
      carg5*(cpkinfo+cempok) +                                         3k2b5
      carg6*(cdumlist),                                                3k2b6
   %results%                                                           3k2c
      cres1*(calostruc) +                                              3k2c1
      cres2*(clistof+cindex+cempok));
                                                                       3k2c2
% trmps  (=> integer) %                                               3k3
   DECLARE dxtrmps = ($xtrmps, 0*cargs + 1*cress,                      3k3a
   %arguments%                                                         3k3b
      0, 0,                                                            3k3b1
   %results%                                                           3k3c
      cres1*(cinteger));
                                                                       3k3c1
% crtchh (psdesc, [struc], index, [index] => index, index) %          3k4
   DECLARE dxcrtchh = ($xcrtchh, 4*cargs + 2*cress,                    3k4a
   %arguments%                                                         3k4b
      carg1*(cpsdesc) +                                                3k4b1
      carg2*(cstruc+cempok) +                                          3k4b2
```

```
        carg3*(cindex),                                          3k4b3
        carg4*(cindex+cempok),                                   3k4b4
    %results%                                                    3k4c
        cres1*(cindex) +                                         3k4c1
        cres2*(cindex));
                                                                 3k4c2
% delchh (index => integer) %                                    3k5
    DECLARE dxdelchh = ($xdelchh, 1*cargs + 1*cress,             3k5a
    %arguments%                                                  3k5b
        carg1*(cindex), 0,                                       3k5b1
    %results%                                                    3k5c
        cres1*(cinteger));
                                                                 3k5c1
% opnpk  (chstlist, [list], index, :list => idxlist) %          3k6
    DECLARE dxopnpk = ($xopnpk, 4*cargs + 1*cress,               3k6a
    %arguments%                                                  3k6b
        carg1*(clistof+ccharstr) +                               3k6b1
        carg2*(clist+cempok) +                                   3k6b2
        carg3*(cindex),                                          3k6b3
        carg4*(cdumlist),                                        3k6b4
    %results%                                                    3k6c
        cres1*(clistof+cindex));
                                                                 3k6c1
% clspk  (idxlist, :list => intlist) %                          3k7
    DECLARE dxclspk = ($xclspk, 2*cargs + 1*cress,               3k7a
    %arguments%                                                  3k7b
```

```
        carg1*(clistof+cindex) +                                    3k7b1
        carg2*(cdumlist), 0,                                        3k7b2
      %results%                                                     3k7c
        cres1*(clistof+cinteger));
                                                                    3k7c1
  %+PDP10% % intpe  (index) %                                       3k8
    DECLARE dxintpe = ($xintpe, 1*cargs + 0*cress,                  3k8a
    %arguments%                                                     3k8b
      carg1*(cindex), 0,                                            3k8b1
    %results%                                                       3k8c
      0);%+PDP10%
                                                                    3k8c1
  %+PDP10% % rsmpe  (index) %                                       3k9
    DECLARE dxrsmpe = ($xrsmpe, 1*cargs + 0*cress,                  3k9a
    %arguments%                                                     3k9b
      carg1*(cindex), 0,                                            3k9b1
    %results%                                                       3k9c
      0);%+PDP10%
                                                                    3k9c1
  %+PDP10% % abrpe  (index) %                                       3k10
    DECLARE dxabrpe = ($xabrpe, 1*cargs + 0*cress,                  3k10a
    %arguments%                                                     3k10b
      carg1*(cindex), 0,                                            3k10b1
    %results%                                                       3k10c
      0);%+PDP10%
                                                                    3k10c1
  % hlppe  (index, index, [struc] => [struc]) %                     3k11
```

```
    DECLARE dxhlppe = (sxhlppe, 3*cargs + 1*cress,                    3k11a
    %arguments%                                                       3k11b
       carg1*(cindex) +                                               3k11b1
       carg2*(cindex) +                                               3k11b2
       carg3*(cstruc+cempok), 0,                                      3k11b3
    %results%                                                         3k11c
       cres1*(calostruc+cempok));
                                                                      3k11c1
%+PDP10% % crtdt  (dsel, [struc], index) %                           3k12
   DECLARE dxcrtdt = (sxcrtdt, 3*cargs + 0*cress,                     3k12a
   %arguments%                                                        3k12b
      carg1*(cdsel) +                                                 3k12b1
      carg2*(cstruc+cempok) +                                         3k12b2
      carg3*(cindex), 0,                                              3k12b3
   %results%                                                          3k12c
      0);%+PDP10%
                                                                      3k12c1
%+PDP10% % deldt  (dsel) %                                           3k13
   DECLARE dxdeldt = (sxdeldt, 1*cargs + 0*cress,                     3k13a
   %arguments%                                                        3k13b
      carg1*(cdsel), 0,                                               3k13b1
   %results%                                                          3k13c
      0);%+PDP10%
                                                                      3k13c1
%+PDP10% % rddt   (dsel => [struc]) %                                3k14
   DECLARE dxrddt = (sxrddt, 1*cargs + 0*cress,                       3k14a
```

```
    %arguments%                                                    3k14b
      carg1*(cdsel), 0,                                            3k14b1
    %results%                                                      3k14c
      cres1*(calostruc+cempok));%+PDP10%                           3k14c1
  %+PDP10% % wrdt   (dsel, [struc]) %                              3k15
    DECLARE dxwrdt = ($xwrdt, 2*cargs + 0*cress,                   3k15a
    %arguments%                                                    3k15b
      carg1*(cdsel) +                                              3k15b1
      carg2*(cstruc+cempok), 0,                                    3k15b2
    %results%                                                      3k15c
      0);%+PDP10%                                                  3k15c1
  %+PDP10% % lckdt  (dsel, index, index, boolean => index) %       3k16
    DECLARE dxlckdt = ($xlckdt, 4*cargs + 1*cress,                 3k16a
    %arguments%                                                    3k16b
      carg1*(cdsel) +                                              3k16b1
      carg2*(cindex) +                                             3k16b2
      carg3*(cindex),                                              3k16b3
      carg4*(cboolean),                                            3k16b4
    %results%                                                      3k16c
      cres1*(cindex));%+PDP10%                                     3k16c1
  %+PDP10% % ulkdt  (dsel, index) %                                3k17
    DECLARE dxulkdt = ($xulkdt, 2*cargs + 0*cress,                 3k17a
    %arguments%                                                    3k17b
```

```
        carg1*(cdsel) +                                          3k17b1
        carg2*(cindex), 0,                                       3k17b2
    %results%                                                    3k17c
        0);%+PDP10%
                                                                 3k17c1
% rdmnu  (=> list) %                                            3k18
    DECLARE dxrdmnu = ($xrdmnu, 0*cargs + 1*cress,               3k18a
    %arguments%                                                  3k18b
        0, 0,                                                    3k18b1
    %results%                                                    3k18c
        cres1*(cstruc));
                                                                 3k18c1
% crtce  (index, struc, index) %                                3k19
    DECLARE dxcrtce = ($xcrtce, 3*cargs + 0*cress,               3k19a
    %arguments%                                                  3k19b
        carg1*(cindex) +                                         3k19b1
        carg2*(cstruc) +                                         3k19b2
        carg3*(cindex), 0,                                       3k19b3
    %results%                                                    3k19c
        0);
                                                                 3k19c1
% alopo  (list, struc, boolean => index, index, struc) %        3k20
    DECLARE dxalopo = ($xalopo, 3*cargs + 3*cress,               3k20a
    %arguments%                                                  3k20b
        carg1*(clist) +                                          3k20b1
        carg2*(cstruc) +                                         3k20b2
```

```
        carg3*(cboolean), 0,                                    3k20b3
    %results%                                                   3k20c
        cres1*(cindex) +                                        3k20c1
        cres2*(cindex) +                                        3k20c2
        cres3*(calostruc));                                     3k20c3
% relpo  (index) %                                             3k21
    DECLARE dxrelpo = (sxrelpo, 1*cargs + 0*cress,              3k21a
    %arguments%                                                 3k21b
        carg1*(cindex), 0,                                      3k21b1
    %results%                                                   3k21c
        0);                                                     3k21c1
% ntepe  (index, index, [struc]) %                             3k22
    DECLARE dxntepe = (sxntepe, 3*cargs + 0*cress,              3k22a
    %arguments%                                                 3k22b
        carg1*(cindex) +                                        3k22b1
        carg2*(cindex) +                                        3k22b2
        carg3*(cstruc+cempok), 0,                               3k22b3
    %results%                                                   3k22c
        0);                                                     3k22c1
% sigpe  (index, [list], [list]) %                             3k23
    DECLARE dxsigpe = (sxsigpe, 3*cargs + 0*cress,              3k23a
    %arguments%                                                 3k23b
        carg1*(cindex) +                                        3k23b1
```

```
            carg2*(clist+cempok) +                                  3k23b2
            carg3*(clist+cempok), 0,                                3k23b3
         %results%                                                  3k23c
            0);
                                                                    3k23c1
      % setrc  (integer) %                                          3k24
         DECLARE dxSetrc = (sxsetrc, 1*cargs + 0*cress,             3k24a
         %arguments%                                                3k24b
            carg1*(cinteger), 0,                                    3k24b1
         %results%                                                  3k24c
            0);
                                                                    3k24c1
   %data structure definition table%                                  31
      % model                                                        311
         0, elmcnt*celms,                                           311a
            celm1*(type) +                                          311a1
            celm2*(type) +                                          311a2
            celm3*(type),                                           311a3
            celm4*(type) +                                          311a4
            celm5*(type) +                                          311a5
            celm6*(type), %
                                                                    311a6
      % dsel   (index, index, charstr, [struc]) %                   312
         DECLARE EXTERNAL ddsdtbl = (0, 4*celms,                    312a
            celm1*(cindex) +                                        312a1
            celm2*(cindex) +                                        312a2
```

```
        celm3*(cucstr),                                          312a3
        celm4*(cstruc+cempok));
                                                                 312a4
    % intpsel (index, index, charstr) %                          313
      DECLARE dintpsel = (0, 3*celms,                            313a
        celm1*(cindex) +                                         313a1
        celm2*(cindex) +                                         313a2
        celm3*(cucstr), 0);
                                                                 313a3
    % extpsel (index, [index], struc) %                          314
      DECLARE dextpsel = (0, 3*celms,                            314a
        celm1*(cindex) +                                         314a1
        celm2*(cindex+cempok) +                                  314a2
        celm3*(cstruc), 0);
                                                                 314a3
    % usrinf (charstr, charstr, charstr) %                       315
      DECLARE dusrinf = ( 0, 3*celms,                            315a
        celm1*(cucstr) +                                         315a1
        celm2*(cucstr) +                                         315a2
        celm3*(cucstr), 0);
                                                                 315a3
    % psloc  (index, index, index) %                            316
      DECLARE dpsloc = (0, 3*celms,                              316a
        celm1*(cindex) +                                         316a1
        celm2*(cindex) +                                         316a2
        celm3*(cindex), 0);
                                                                 316a3
```

```
    % psdesc (charstr, charstr, psloc, [charstr]) %                    317
       DECLARE dpsdesc = (0, 4*celms,                                  317a
          celm1*(ccharstr) +                                           317a1
          celm2*(ccharstr) +                                           317a2
          celm3*(cpsloc),                                              317a3
          celm4*(ccharstr+cempok));                                    317a4
    % abrinfo (integer, [charstr]) %                                   318
       DECLARE dabrinfo = (0, 2*celms,                                 318a
          celm1*(cinteger) +                                           318a1
          celm2*(ccharstr+cempok), 0);                                 318a2
    % pkinfo (list, [list], index) %                                   319
       DECLARE dpkinfo = (0, 3*celms,                                  319a
          celm1*(clistof+ccharstr) +                                   319a1
          celm2*(clist+cempok) +                                       319a2
          celm3*(cindex), 0);                                          319a3
%IPC definition table%                                                3m
   DECLARE EXTERNAL dipctbl = (                                       3m1
   %inter-fork%                                                       3m2
      $fcrtps, $fdelps, $fcrtce, $fdelce,                             3m2a
      $falopo, $frelpo, $fsndch, $frcvch, $fupdpo,                    3m2b
   %inter-host%                                                       3m3
      $ncrtps, $ndelps, $ncrtce, $ndelce,                             3m3a
      $nalopo, $nrelpo, $nsndch, $nrcvch, $nupdpo);                   3m3b
```

```
   %miscellaneous%                                                    3n
      %system SAV filenames% DECLARE EXTERNAL STRING                  3n1
         dpsavf  =     "DPS.SAV", %DPS itself%                        3n1a
         dencap  =   "ENCAP.SAV"; %encapsulator%                      3n1b
      %miscellaneous% DECLARE EXTERNAL                                3n2
         dlhostn = 545057636456B, %SIXBIT LHOSTN table name%          3n2a
         dermsgs =          TRUE, %error message table switch%        3n2b
         dtrace  =            -1, %trace switch%                      3n2c
         dpatch        [cmptchl], %patch space%                       3n2d
         dtrptbl = (0,0,0,0,0,0,0,2B10,0,0,0,0,0,0,0);                3n2e
            %JSYS trap table (trapping JSYS 400)%                     3n2e1
   FINISH                                                             3o
FILE dpsdta % compiler (xl10,)  output to (dpsdta.rel,) %             4
   %pre-compile switches%                                             4a
      NOMESS;                                                         4a1
      SET PDP10 = TRUE;                                               4a2
   %assure XL10DATA and DPSDTA share no page%                         4b
      DECLARE padding [511];                                          4b1
   %declarations%                                                     4c
      %global constants%                                              4c1
         DECLARE EXTERNAL CONSTANT                                    4c1a
         cmctxl  =   5, %manager context length%                      4c1b
         cmpsnl  =  40, %max length of process name%                  4c1c
         cmerml  = 100, %max length of error message%                 4c1d
         cmlkskl = 100, %length of lock stack%                        4c1e
```

```
        cmwtdel =  10, %length of watchdog ECB%                    4c1f
        cwnmpl  =   2; %length (in words) of window map%           4c1g
        DECLARE CONSTANT                                           4c1h
        cll10r  = 13B, %length in pages of XL10DATA%               4c1i
        cnoacs  =  16; %number of ACs%                             4c1j
    %folder addresses%                                             4d
      DECLARE EXTERNAL variables [0],                              4d1
      vcafld,          %addr of calls folder%                      4d2
      vcnfld,          %addr of channels folder%                   4d3
      vdtfld,          %addr of data stores folder%                4d4
      vevfld,          %addr of events folder%                     4d5
      vlkfld,          %addr of locks folder%                      4d6
      vlsfld,          %addr of lock sets folder%                  4d7
      vmgfld,          %addr of managers folder%                   4d8
      vpkfld,          %addr of packages folder%                   4d9
      vpofld,          %addr of ports folder%                      4d10
      vprfld,          %addr of processors folder%                 4d11
      vpsfld,          %addr of processes folder%                  4d12
      vsgfld,          %addr of segments folder%                   4d13
      vsufld,          %addr of subprocesses folder%               4d14
      vsyfld,          %addr of system calls folder%               4d15
      vusfld,          %addr of user calls folder%                 4d16
      vtmfld,          %addr of timers folder%                     4d17
    %scheduler parameters%                                         4e
      vmgroot,         %addr of top procedure%                     4e1
```

```
    vnxtmg,         %addr of nxt manager record%                    4e2
    vmg,            %addr of cur manager record%                    4e3
    vnewmg,         %addr of new manager record%                    4e4
    vmgh,           %cur manager handle%                            4e5
    vnewh,          %new manager handle%                            4e6
    vacs,           %addr of cur manager ACs%                       4e7
    vnewacs,        %addr of new manager ACs%                       4e8
    vnewbase,       %page number of new manager state%              4e9
    vindex,         %ECB list index%                                4e10
    vcode,          %event completion code%                         4e11
    vlength,        %total number of completions%                   4e12
%DPS state%                                                         4f
    vstate [0],     %state:%                                        4f1
      vlckstk [cmlkskl], %lock stack%                               4f1a
      vctx [0],       %context (see also CMCTXL):%                  4f1b
        vcch,           %controlling/chaining call handle%          4f1b1
        vcpcrid [0],        %processor id:%                         4f1b2
          vcph,           %process handle%                          4f1b2a
          vcsph,          %subprocess handle%                       4f1b2b
          vcpcrh,         %processor handle%                        4f1b2c
        vcspldr,        %subprocess leader's PCRH%                  4f1b3
      vwheel,         %wheel?%                                      4f1c
      Vwhlpls,        %TRUE if RUNFLD to run as wheel%              4f1d
      vdgmsg;         %addr for signal diagnostic message%          4f1e
      DECLARE EXTERNAL STRING                                       4f1f
```

```
        verrmsg [cmerml]; %error message%                          4f1g
    DECLARE EXTERNAL                                               4f2
        vstalen [0];   %end of state%                              4f2a
%information about self%                                           4g
    DECLARE EXTERNAL                                               4g1
    vfpsdesc,       %addr of process description%                  4g2
    vtrldr,         %TRUE (1) if tree leader%                      4g3
    vfph,           %absolute process handle%                      4g4
    vfhost,         %local host addr%                              4g5
    vfjob,          %local job number%                             4g6
    vpsldr,         %process leader's SPH%                         4g7
    vsignin,        %TRUE if INIPS called by superior%             4g8
    vsplok,         %TRUE if process can be spliced to%            4g9
    vdrain,         %TRUE if system shutdown in progress%          4g10
    vsph,           %superior's absolute process handle%           4g11
    vxcost;         %cost of system procedures%                    4g12
    DECLARE EXTERNAL STRING                                        4g13
    vpsname [cmpsnl]; %process name%                               4g14
%PSI parameters%                                                   4h
    DECLARE EXTERNAL                                               4h1
    vwtdecb [cmwtdel+1], %PSI ECB%                                 4h2
    vwtdcde,        %PSI ECB code%                                 4h3
    vpsipc,         %save area for PSI PC%                         4h4
    vpsiacs [cnoacs], %save area for PSI ACs%                      4h5
    vwaitpc,        %PSI activate-DPS PC%                          4h6
```

```
    vchntab [36],   %PSI channel table%                          4h7
    vlevtab [3],    %PSI level table%                            4h8
    villpc,         %illegal instruction PC%                     4h9
%timer parameters%                                               4i
    vtmrfkh,        %timer fork handle%                          4i1
    vtmrintv,       %current interval in ms%                     4i2
    vtmrstrt,       %starting time of current interval%          4i3
%meters%                                                         4j
    vmoprc,         %# operations (OPRMGR)%                      4j1
    vmvjuc,         %# VJUSRs (VJUSR)%                           4j2
    vmvjsc,         %# VJSYSs (IVDPS)%                           4j3
    vmsndc,         %# messages sent (SNDPS)%                    4j4
    vmrcvc,         %# messages received (RCVPS)%                4j5
    vmmgrc,         %# managers spawned (SPAWN)%                 4j6
    vmcrfc,         %# records created (CRTREC)%                 4j7
    vmrnfc,         %# folder funs (RUNFLD)%                     4j8
    vmflkc,         %# locks set fast (SETLCB)%                  4j9
    vmslkc,         %# locks set slow (SETLCB)%                  4j10
    vmwlkc,         %# locks set after wait (SETLCB)%            4j11
    vmsigc,         %# ECB signals (SIGECB)%                     4j12
    vmvwtc,         %# virtual waits (WAIECB)%                   4j13
    vmrwtc,         %# real waits (OWTSIG)%                      4j14
    vmderc,         %# DPS errors (SIGERR)%                      4j15
    vmlerc,         %# L10 errors (ERR)%                         4j16
    vmoerc,         %# operating system errors (OSIGERR)%        4j17
```

```
    %misc%                                                          4k
      vgtjtbl [10B], %long GTJFN table%                             4k1
      vfmsg,         %addr of message from self%                    4k2
      vwinbas,       %addr of B36 base port window%                 4k3
      vchmnu,        %addr of B36 channel menu%                     4k4
      vwnmp [cwnmpl], %inter-fork window map%                       4k5
      vl10jfn,       %L10 state file JFN%                           4k6
      vstgzon,       %addr of storage zone%                         4k7
      varend [0],    %end of variables%                             4k8
    %not-to-be-zeroed variables%                                    41
      vfsvhnd  [cl110r], %handles for XL10DATA src pages%           411
      vstupacs [cnoacs]; %startup ACs; not to be zeroed%            412
        % R0     = XWD channel type, format %                       412a
        % R1-R15 = ASCIZ intrah0staddr if tree leader %             412b
    FINISH                                                          4m
FILE dpstrc % compiler (xl10,)  output to (dpstrc,rel,) %           5
    %pre-compile switches%                                         5a
      NOMESS;                                                       5a1
      SET PDP10 = TRUE;                                             5a2
    %declarations%                                                  5b
      %global constants%                                            5b1
        DECLARE CONSTANT                                            5b1a
        %parameter list types%                                      5b1b
          carglist = 1, %argument list%                             5b1b1
          creslist = 2, %result list%                               5b1b2
```

```
        cabrlist = 3, %abort list%                                  5b1b3

      %message transmission modes%                                  5b1c

        csend    = 5, %send%                                        5b1c1

        creceive = 6, %receive%                                     5b1c2

      %manager trace types%                                         5b1d

        chold    = 1, %held%                                        5b1d1

        crelease = 2, %released%                                    5b1d2

      %miscellaneous%                                               5b1e

        cmnumb   = 2; %msg number element number%                   5b1e1

      DECLARE EXTERNAL CONSTANT                                     5b1f

      %user/system call types%                                      5b1g

        cusc     = 1, %user call%                                   5b1g1

        csyc     = 2; %system call%                                 5b1g2

%tracers%                                                           5c

  (trcmg) %trace manager%
  PROCEDURE (mode, mgh);                                            5c1

    %declarations%                                                  5c1a

      LOCAL STRING text [40];                                       5c1a1

      LOCAL action REF;                                             5c1a2

    %compose trace entry%                                           5c1b

      &action _ IF mode = chold THEN $" held" ELSE $"
      released";                                                    5c1b1

      *text* _                                                      5c1b2

        "REL MGH=", STRING (mgh-cmgmnh+1), *action*, " by lock
        management";                                                5c1b2a

    %output trace entry%                                            5c1c
```

```
        oprtstr (stext);                                        5c1c1
    %return%                                                    5c1d
        RETURN;                                                 5c1d1
        END.                                                    5c1d2

(trcpo) %trace message%
PROCEDURE (mode, poh, block REF);                               5c2
    %declarations%                                              5c2a
        LOCAL number;                                           5c2a1
        LOCAL STRING text [500], port [40];                     5c2a2
        LOCAL LIST msg [20];                                    5c2a3
        LOCAL dir REF;                                          5c2a4
    %catchphrases%                                              5c2b
        (pnulist) CATCHPHRASE; IF abr () THEN NULL-LISTS;       5c2b1
    %decode message number%                                     5c2c
        decstruc (cmessage, &block, $msg);                      5c2c1
        INVOKE (pnulist);                                       5c2c2
        number _ ELEM #msg# [cmnumb];                           5c2c3
    %compose trace entry%                                       5c2d
        &dir _ CASE mode OF                                     5c2d1
            = csend:    $"to ";                                 5c2d1a
            = creceive: $"from ";                               5c2d1b
            ENDCASE     $"from and to ";                        5c2d1c
        IF poh THEN *port* _ "POH=", STRING (poh)               5c2d2
        ELSE *port* _ "SELF";                                   5c2d3
        *text* _                                                5c2d4
```

```
        "MSG ", *[dmmneum [number-1]]*, " (", STRING (number),
        ") ", *dir*, *port*, ':, EOL, "    ";                    5c2d4a

      porstruc (&block, $text);                                   5c2d5

    %output trace entry%                                          5c2e

      oprtstr ($text);                                            5c2e1

    %return%                                                      5c2f

      RETURN;                                                     5c2f1

      END.
                                                                  5c2f2

(trcpr) %trace user/system call arguments/results%
PROCEDURE (calltype, number, entry POINTER, prmltype, prml
REF);                                                             5c3

    %declarations%                                                5c3a

      LOCAL cnt, specs, block, dst;                               5c3a1

      LOCAL STRING text [500];                                    5c3a2

      LOCAL callt REF, prmlt REF, dir REF, mneumonics REF;        5c3a3

    %encode parameter list%                                       5c3b

      CASE prmltype OF                                            5c3b1

        = carglist;                                               5c3b1a

          BEGIN                                                   5c3b1a1

          cnt    _ entry.enargc;                                  5c3b1a2

          specs  _ entry+2;                                       5c3b1a3

          &prmlt _ $"argl ";                                      5c3b1a4

          END;                                                    5c3b1a5

        = creslist;                                               5c3b1b

          BEGIN                                                   5c3b1b1

          cnt    _ entry.enresc;                                  5c3b1b2
```

```
            specs  _ entry + (IF calltype = cusc THEN 4 ELSE
            5);                                                    5c3b1b3
            &prmlt _ $"resl ";                                     5c3b1b4
            END;                                                   5c3b1b5
         ENDCASE                                                   5c3b1c
            &prmlt _ $"abrl ";                                     5c3b1c1
      IF prmltype = cabrlist THEN                                  5c3b2
         block _ encstruc (cabrinfo, &prml)                        5c3b2a
      ELSE                                                         5c3b3
         BEGIN                                                     5c3b3a
         dst _ aloent (clist, prml,L);                             5c3b3b
         INVOKE (palwrl,, dst);                                    5c3b3c
         block _                                                   5c3b3d
            encstruc (clist, encprms (&prml, cnt, specs, dst));
                                                                   5c3b3d1
         END;                                                      5c3b3e
      INVOKE (palwrb,, block);                                     5c3b4
   %compose trace entry%                                           5c3c
      IF calltype = cusc THEN                                      5c3c1
         BEGIN                                                     5c3c1a
         &callt      _ $"VJUSR ";                                  5c3c1b
         &mneumonics _ $dumneum;                                   5c3c1c
         END                                                       5c3c1d
      ELSE                                                         5c3c2
         BEGIN                                                     5c3c2a
         &callt      _ $"VJSYS ";                                  5c3c2b
```

```
            &mneumonics _ $dsmneum;                                    5c3c2c

            END;                                                       5c3c2d

         &dir   _                                                      5c3c3

            IF (calltype = cusc AND prmltype = carglist) OR
            (calltype = csyc AND prmltype # carglist) THEN $"to "
            ELSE $"from ";                                             5c3c3a

         *text* _                                                      5c3c4

            *callt*, *[mneumonics [number-1]]*, " (", STRING
            (number), ") ", *prmlt*, *dir*, "PCRH=", STRING
            (vcpcrh), ':, EOL, "    ";                                 5c3c4a

         porstruc (block, $text);                                      5c3c5

      %output trace entry%                                             5c3d

         oprtstr ($text);                                              5c3d1

      %return%                                                         5c3e

         RETURN;                                                       5c3e1

         END.
                                                                       5c3e2

   (trcerr) %trace error%
   PROCEDURE (number, msg REF);                                        5c4

      %declarations%                                                   5c4a

         LOCAL STRING text [100];                                      5c4a1

      %compose trace entry%                                            5c4b

         *text* _ "ERROR ", STRING (number), " noted: ", *msg*;        5c4b1

      %output trace entry%                                             5c4c

         oprtstr ($text);                                              5c4c1

      %return%                                                         5c4d

         RETURN;                                                       5c4d1
```

```
        END,                                                    5c4d2

%utilities%                                                     5d

    (porstruc) %portray data structure%
    PROCEDURE (src REF, dst REF %=> dst REF%);                  5d1

        %declarations%                                          5d1a

            LOCAL len, i, nxtsrc, wrdcnt, overhang;             5d1a1

            LOCAL STRING nxtbits [(WORD+2)/3];                  5d1a2

        %note key%                                              5d1b

            IF src.dskey THEN                                   5d1b1

                BEGIN                                           5d1b1a

                *dst* _ *dst*, '[;                              5d1b1b

                sizstruc (&src : len);                          5d1b1c

                porstruc (&src+len, &dst);                      5d1b1d

                *dst* _ *dst*, '];                              5d1b1e

                END;                                            5d1b1f

        %dispatch on basis of data type%                        5d1c

            len _ src.dslen;                                    5d1c1

            CASE src.dstype OF                                  5d1c2

                = cempty:                                       5d1c2a

                    *dst* _ *dst*, 'E;                          5d1c2a1

                = cboolean:                                     5d1c2b

                    *dst* _ *dst*, 'B, STRING (len);            5d1c2b1

                = cindex:                                       5d1c2c

                    *dst* _ *dst*, 'H, STRING (len);            5d1c2c1

                = cinteger:                                     5d1c2d
```

```
        BEGIN                                                   5d1c2d1
        nxtsrc _ src [1];                                       5d1c2d2
        IF nxtsrc,LH THEN                                       5d1c2d3
           *dst* _ *dst*, "I(", STRING (nxtsrc,LH, 8), ',,
           STRING (nxtsrc,RH, 8), ')                            5d1c2d3a
        ELSE                                                    5d1c2d4
           *dst* _ *dst*, 'I, STRING (nxtsrc, 8);               5d1c2d4a
        END;                                                    5d1c2d5
      = cbitstr:                                                5d1c2e
        BEGIN                                                   5d1c2e1
        *dst* _ *dst*, 'A, STRING (len), '";                    5d1c2e2
        wrdcnt _ (len+WORD-1)/WORD;                             5d1c2e3
        FOR i _ 1 UP UNTIL > wrdcnt DO                          5d1c2e4
           BEGIN                                                5d1c2e4a
           *nxtbits* _ STRING (src [i], 8);                     5d1c2e4b
           WHILE nxtbits,L < WORD/3 DO                          5d1c2e4c
              *nxtbits* _ '0, *nxtbits*;                        5d1c2e4c1
           *dst* _ *dst*, *nxtbits*;                            5d1c2e4d
           END;                                                 5d1c2e4e
        IF overhang _ (len MOD WORD) THEN                       5d1c2e5
           dst,L _ dst,L - (WORD - overhang)/3;                 5d1c2e5a
        *dst* _ *dst*, '";                                      5d1c2e6
        END;                                                    5d1c2e7
      = ccharstr:                                               5d1c2f
        *dst* _ *dst*, 'C, '", *src*, '";                       5d1c2f1
```

```
            = clist;                                                    5d1c2g
               BEGIN                                                    5d1c2g1
               *dst* _ *dst*, "L(";                                     5d1c2g2
               nxtsrc _ &src + 1;                                       5d1c2g3
               FOR i _ 1 UP UNTIL > len DO                              5d1c2g4
                  BEGIN                                                 5d1c2g4a
                  IF i > 1 THEN *dst* _ *dst*, ',;                      5d1c2g4b
                  porstruc (nxtsrc := nxtsrc + sizstruc (nxtsrc),
                  &dst);                                                5d1c2g4c
                  END;                                                  5d1c2g4d
               *dst* _ *dst*, ');                                       5d1c2g5
               END;                                                     5d1c2g6
            ENDCASE *dst* _ *dst*, '?;                                  5d1c2h
      %return%                                                          5d1d
         RETURN (&dst);                                                 5d1d1
         END,
                                                                        5d1d2
%mneumonics%                                                            5e
   %system calls%                                                       5e1
      DECLARE EXTERNAL dsmneum = (                                      5e1a
         $"crtps", $"delps", $"itdps", $"sepps",                        5e1a1
         $"opnpk", $"clspk", $"calpe", $"vispe",                        5e1a2
         $"aloch", $"relch", $"acqpe", $"relpe",                        5e1a3
         $"intpe", $"rsmpe", $"ntepe", $"hlppe",                        5e1a4
         $"crtdt", $"deldt", $"rddt" , $"wrdt" ,                        5e1a5
         $"lckdt", $"ulkdt", $"crtch", $"delch",                        5e1a6
```

```
            $"crtsp", $"delsp", $"crtpr", $"delpr",                 5e1a7
            $"sipr" , $"rdypr", $"sndch", $"rcvch",                 5e1a8
            $"crtlk", $"dellk", $"setlk", $"remlk",                 5e1a9
            $"crtev", $"delev", $"sigev", $"tstev",                 5e1a10
            $"waiev", $"infps");                                    5e1a11
        %user calls%                                                5e2
          DECLARE EXTERNAL dumneum = (                              5e2a
            $"sopr" , $"inipk", $"trmpk", $"pecal",                 5e2a1
            $"peint", $"persm", $"peabr", $"pente",                 5e2a2
            $"pehlp", $"lvrdt", $"lrddt", $"lwrdt",                 5e2a3
            $"okips", $"oksps", $"okopk", $"okcpk",                 5e2a4
            $"okcch", $"okdch", $"ntlch", $"ntdps");                5e2a5
        %messages%                                                  5e3
          DECLARE EXTERNAL dmmneum = (                              5e3a
            $"calpe", $"recpe", $"rtnpe", $"ntepe",                 5e3a1
            $"error");                                              5e3a2
    %error messages%                                                5f
      %dispatch table%                                              5f1
        DECLARE EXTERNAL dertbl = (13,                              5f1a
          $ddermsgs, $deermsgs, $dvermsgs, $dfermsgs,               5f1a1
          $dcermsgs, $dlermsgs, $dkermsgs, $dpermsgs,               5f1a2
          $dsermsgs, $drermsgs, $dmermsgs, $dgermsgs,               5f1a3
          $dhermsgs);                                               5f1a4
      %data structures%                                             5f2
        DECLARE ddermsgs = (20,                                     5f2a
```

```
        %edddst =   1% $"?,",                                          5f2b
        %eddkey =   2% $"Duplicate key,",                              5f2c
        %eddrky =   3% $"Key has key,",                                5f2d
        %eddsto =   4% $"Duplicate data store name,",                  5f2e
        %ediidx =   5% $"Illegal INDEX,",                              5f2f
        %edipdl =   6% $"Illegal PSEL/DSEL,",                          5f2g
        %ediuif =   7% $"Illegal USERINFO,",                           5f2h
        %edmkey =   8% $"Missing key,",                                5f2i
        %edodcl =   9% $"LIST too long to decode,",                    5f2j
        %edolst =  10% $"Maximum LIST size exceeded,",                 5f2k
        %edostr =  11% $"CHRSTR too long to decode,",                  5f2l
        %edufty =  12% $"Undefined data type,",                        5f2m
        %eduity =  13% $"Undefined informal data type,",               5f2n
        %edusto =  14% $"Undefined data store name,",                  5f2o
        %edwesl =  15% $"Non-LIST addressed by ESEL,",                 5f2p
        %edwidx =  16% $"No such index,",                              5f2q
        %edwkey =  17% $"No such key,",                                5f2r
        %edwtyp =  18% $"Wrong data type,",                            5f2s
        %edwpmc =  19% $"Incorrect number of parameters,",             5f2t
        %edoabc =  20% $"Data structure overflows source block,");     5f2u
    %errors%                                                           5f3
        DECLARE deermsgs = (5,                                         5f3a
        %eefimp =  51% $"Not implemented,",                            5f3b
        %eefops =  52% $"Operating system error,",                     5f3c
        %eemerr =  53% $"Unidentifiable operating system error,",      5f3d
```

```
        %eeuern =  54% $"Undefined error number,",                    5f3e
        %eefl10 =  55% $"L10 run-time error,");                       5f3f
    %events%                                                          5f4
        DECLARE dvermsgs = (5,                                        5f4a
        %evfacq = 101% $"Won't delete ALOCH event,",                  5f4b
        %evfjuv = 102% $"Won't delete SIPR event,",                   5f4c
        %evilen = 103% $"Illegal event length,",                      5f4d
        %evoecb = 104% $"Event overflow,",                            5f4e
        %evftmr = 105% $"Won't delete SETMR event,");                 5f4f
    %folders%                                                         5f5
        DECLARE dfermsgs = (4,                                        5f5a
        %efodrn = 151% $"Won't create record while folder drained,",  5f5b
        %eforun = 152% $"RUNFLD overrun,",                            5f5c
        %efuift = 153% $"Undefined record information type,",         5f5d
        %efurop = 154% $"Undefined RUNFLD operation,");               5f5e
    %inter-process communication%                                     5f6
        DECLARE dcermsgs = (4,                                        5f6a
        %ecdcce = 201% $"Channel already created,",                   5f6b
        %ecutyp = 202% $"Undefined channel type,",                    5f6c
        %ecwmnu = 203% $"Channel type menu mismatch,",                5f6d
        %ecwpkl = 204% $"Inconsistent packet length,");               5f6e
    %locks%                                                           5f7
        DECLARE dlermsgs = (7,                                        5f7a
        %elfded = 251% $"Deadlock,",                                  5f7b
        %elfdel = 252% $"Sought lock deleted,",                       5f7c
```

```
        %elflck = 253% $"Lock attempt failed,",                          5f7d
        %elmswp = 254% $"Non-existent LCB to be swapped,",               5f7e
        %elmstk = 255% $"Lock stack underflow,",                         5f7f
        %elostk = 256% $"Lock stack overflow,",                          5f7g
        %elistk = 257% $"Lock stack surplus,");                          5f7h
    %packages%                                                           5f8
        DECLARE dkermsgs = (2,                                           5f8a
        %ekfded = 301% $"Package dead,",                                 5f8b
        %ekupkn = 302% $"Undefined package,");                           5f8c
    %procedures%                                                         5f9
        DECLARE dpermsgs = (26,                                          5f9a
        %epfnoh = 351% $"No help available,",                            5f9b
        %epfpio = 352% $"No processor with sufficient priority,",        5f9c
        %epfsab = 353% $"Won't abort system procedure,",                 5f9d
        %epfsin = 354% $"Won't interrupt system procedure,",             5f9e
        %epiacq = 355% $"Context prohibits ACQPE,",                      5f9f
        %epiaid = 356% $"Context prohibits any action by local
        procedure,",                                                     5f9g
        %epihlp = 357% $"Context prohibits HLPPE,",                      5f9h
        %epiint = 358% $"Context prohibits INTPE,",                      5f9i
        %epimsk = 359% $"Illegal argument/result list mask,",            5f9j
        %epinte = 360% $"Context prohibits NTEPE,",                      5f9k
        %epiotc = 361% $"Illegal system procedure outcome,",             5f9l
        %epirel = 362% $"Context prohibits RELPE,",                      5f9m
        %epirsm = 363% $"Context prohibits RSMPE,",                      5f9n
```

```
        %epixhp = 364% $"Context prohibits call to XHLPPE.",          5f9o
        %epixin = 365% $"Context prohibits call to XINTPE.",          5f9p
        %epixnt = 366% $"Context prohibits sending XNTEPE.",          5f9q
        %epixrc = 367% $"Context prohibits sending XRECPE.",          5f9r
        %epixrm = 368% $"Context prohibits call to XRSMPE.",          5f9s
        %epixrn = 369% $"Context prohibits sending XRTNPE.",          5f9t
        %epuotc = 370% $"Undefined procedure outcome.",               5f9u
        %epurtn = 371% $"Undefined return type.",                     5f9v
        %epusyn = 372% $"Undefined system procedure number.",         5f9w
        %epwvis = 373% $"Unplanned for visit.",                       5f9x
        %epfqin = 374% $"Procedure can't be interrupted.",            5f9y
        %episig = 375% $"Context prohibits SIGPE.",                   5f9z
        %epixsg = 376% $"Context prohibits call to XSIGPE.");         5f9a@
    %processes%                                                       5f10
        DECLARE dsermsgs = (10,                                       5f10a
        %esdpoh = 401% $"POH already associated with process.",       5f10b
        %esfded = 402% $"Process dead.",                              5f10c
        %esipsa = 403% $"Syntax error in process addr.",              5f10d
        %esircv = 404% $"No POH via which to receive message.",       5f10e
        %esisup = 405% $"Not direct superior.",                       5f10f
        %esmpoh = 406% $"No POH via which to send message.",          5f10g
        %esumsg = 407% $"Undefined message number.",                  5f10h
        %esuser = 408% $"Undefined user name.",                       5f10i
        %esuwtd = 409% $"Undefined watchdog code.",                   5f10j
        %eswqak = 410% $"?,");                                        5f10k
```

```
%processors%                                                    5f11
   DECLARE drermsgs = (18,                                      5f11a
   %erigtd = 451% $"Context prohibits GTDPS.",                  5f11b
   %eriptd = 452% $"Context prohibits PTDPS.",                  5f11c
   %erisin = 453% $"Not signed in.",                            5f11d
   %erorsb = 454% $"ABF overflow.",                             5f11e
   %erosml = 455% $"Small block overflow.",                     5f11f
   %erowin = 456% $"Processor window overflow.",                5f11g
   %erualo = 457% $"Undefined entity type allocated.",          5f11h
   %eruinf = 458% $"Undefined process information type.",       5f11i
   %eruopn = 459% $"Undefined operation number.",               5f11j
   %erupml = 460% $"Undefined parameter location.",             5f11k
   %erupsi = 461% $"Undefined PSI channel number.",             5f11l
   %erurde = 462% $"Undefined entity type read.",               5f11m
   %eruscp = 463% $"Undefined scope.",                          5f11n
   %erusyc = 464% $"Undefined system call number.",             5f11o
   %eruusc = 465% $"Undefined user call number.",               5f11p
   %eruwre = 466% $"Undefined entity type written.",            5f11q
   %erfded = 467% $"Processor dead.",                           5f11r
   %erdsin = 468% $"Processor already signed in.");             5f11s
%storage%                                                       5f12
   DECLARE dmermsgs = (5,                                       5f12a
   %emfexh = 501% $"CF storage exhausted.",                     5f12b
   %emient = 502% $"Negative entity size.",                     5f12c
   %emuent = 503% $"Undefined entity type.",                    5f12d
```

```
    %emibyp = 504% $"Illegal user-supplied byte pointer.",          5f12e
    %emiadr = 505% $"Illegal user-supplied address.");              5f12f
  %undefined handles%                                               5f13
    DECLARE dgermsgs = (16,                                         5f13a
    %egmhca = 551% $"Undefined call handle.",                       5f13b
    %egmhcn = 552% $"Undefined channel handle.",                    5f13c
    %egmhdt = 553% $"Undefined data store handle.",                 5f13d
    %egmhev = 554% $"Undefined event handle.",                      5f13e
    %egmhlk = 555% $"Undefined lock handle.",                       5f13f
    %egmhls = 556% $"Undefined lockset handle.",                    5f13g
    %egmhmg = 557% $"Undefined manager handle.",                    5f13h
    %egmhpk = 558% $"Undefined package handle.",                    5f13i
    %egmhpo = 559% $"Undefined port handle.",                       5f13j
    %egmhpr = 560% $"Undefined processor handle.",                  5f13k
    %egmhps = 561% $"Undefined process handle.",                    5f13l
    %egmhsg = 562% $"Undefined segment handle.",                    5f13m
    %egmhsu = 563% $"Undefined subprocess handle.",                 5f13n
    %egmhsy = 564% $"Undefined system call handle.",                5f13o
    %egmhus = 565% $"Undefined user call handle.",                  5f13p
    %egmhtm = 566% $"Undefined timer handle.");                     5f13q
  %no more handles%                                                 5f14
    DECLARE dhermsgs = (16,                                         5f14a
    %ehohca = 601% $"No call handle available.",                    5f14b
    %ehohcn = 602% $"No channel handle available.",                 5f14c
    %ehohdt = 603% $"No data store handle available.",              5f14d
```

```
        %ehohev = 604% $"No event handle available.",                5f14e
        %ehohlk = 605% $"No lock handle available.",                 5f14f
        %ehohls = 606% $"No lockset handle available.",              5f14g
        %ehohmg = 607% $"No manager handle available.",              5f14h
        %ehohpk = 608% $"No package handle available.",              5f14i
        %ehohpo = 609% $"No port handle available.",                 5f14j
        %ehohpr = 610% $"No processor handle available.",            5f14k
        %ehohps = 611% $"No process handle available.",              5f14l
        %ehohsg = 612% $"No segment handle available.",              5f14m
        %ehohsu = 613% $"No subprocess handle available.",           5f14n
        %ehohsy = 614% $"No system call handle available.",          5f14o
        %ehohus = 615% $"No user call handle available.",            5f14p
        %ehohtm = 616% $"No timer handle available.");               5f14q
    FINISH                                                           5g
```

DPS-10 Version 2,5 Source Code

(J26267) 13-AUG-75 14:08;;;; Title: Author(s): James E. (Jim) White/JEW; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JEW; Origin: < JWHITE, C2DPS,NLS;19, >, 13-AUG-75 13:59 JEW ;;;;####;

26267 Distribution

Douglas C. Engelbart, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, James C. Norton, Jeffrey C. Peters, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Richard W. Watson, Don I. Andrews, Mary Ann Kellan, Buddie J. Pine, Andy Poggio, David L. Retz, Laura J. Metzger, Karolyn J. Martin, Jan A. Cornish, Larry L. Garlick, Priscilla A. Wold, Pamela K. Allen, Delorse M. Brooks, Beverly Boli, Rita Hysmith, Log Augmentation, Joseph L. Ehardt, Raymond R. Panko, Susan Gail Roetter, Robert Louis Belleville, Rene C. Ochoa, Ann Weinberg, Joan Hamilton, Adrian C. McGinnis, Robert S. Ratner, David S. Maynard, Robert N. Lieberman, Sandy L. Johnson, James H. Bair, Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Marcia L. Keeney, Elizabeth K. Michael, Jonathan B. Postel, Elizabeth J. Feinler, Kirk E. Kelley, N. Dean Meyer, James E. (Jim) White