JEW 13-AUG-75 14:08 26267

DPS-10 Version 2,5 Source Code

Think hard before you print this 170-page document. This is the L10 source code for the version (2,5) of DPS-10 in existence when DPS was cut from NSW. This program compiles, loads, and executes successfully.

DPS-10 Version 2.5 Source Code

```
FILE dps    % compiler (x110,)  output to    (dps.rel,) %                    1
  %pre-compile switches%                                                    1a
    NOMESS;                                                                 1a1
    SET PDP10 = TRUE;                                                       1a2
  %declarations%                                                            1b
    %local pointers% POINTER                                                1b1
      vnxtmg,  %addr of nxt manager's record%                               1b1a
      vmg,     %addr of cur manager's record%                               1b1b
      vnewmg;  %addr of new manager's record%                               1b1c
    %global constants%                                                      1b2
      DECLARE EXTERNAL CONSTANT                                             1b2a
      cok      =1B5; %system OK completion code%                            1b2b
      DECLARE CONSTANT                                                      1b2c
      cpglen   =512, %length (in words) of page%                            1b2d
      chndwid =  6, %local handle width%                                    1b2e
      cmpsnl  = 40, %max length of process name%                            1b2f
      cmhsnl  = 40, %max length of host name%                               1b2g
      cmlkidl =  3, %max length of lock id%                                 1b2h
      cmctxl  =  5, %length of manager context%                             1b2i
      cmpidl  =  3, %length of processor id%                                1b2j
      cnoacs  = 16; %number of ACs%                                         1b2k
    %local constants% DECLARE CONSTANT                                      1b3
      %universal user interface dependent%                                  1b3a
        %max list lengths%                                                  1b3a1
          cmlistl     =   64, %any list%                                    1b3a1a
```

```
        cmdumll     =   64, %dummy list%                    1b3a1b
        cmpkl       =   20, %package name list%             1b3a1c
        cmpsell     =    3, %pselector%                     1b3a1d
        cmdsell     =    4, %dselector%                     1b3a1e
        cmesell     =   64, %eselector%                     1b3a1f
        cmusrinfl   =    3, %user information%              1b3a1g
    %max character string lengths%                          1b3a2
        cmpsal      =   40, %process address%               1b3a2a
        cmactl      =    3, %action%                        1b3a2b
        cmiahal     =   40, %intra-host address%            1b3a2c
        cmpknl      =   40, %package name%                  1b3a2d
        cmpenl      =   40, %procedure name%                1b3a2e
    %procedure outcomes%                                    1b3a3
        cvisit      =    1, %visiting%                      1b3a3a
        csuccess    =    2, %success%                       1b3a3b
        cfailure    =    3, %failure%                       1b3a3c
        caborted    =    4, %aborted%                       1b3a3d
        csiged      =    5, %signalled%                     1b3a3e
    %argument list mask values%                             1b3a4
        ccaller     =    1, %to/from caller%                1b3a4a
        cdiscard    =    2, %discard%                       1b3a4b
    %lock types%                                            1b3a5
        cshare      =    1, %share%                         1b3a5a
        cexclusive  =    2, %exclusive%                     1b3a5b
    %scopes%                                                1b3a6
```

```
        cprocessor  =    1, %processor%                          1b3a6a
        csubprocess =    2, %subprocess%                         1b3a6b
        cprocess    =    3, %process%                            1b3a6c
        call        =    4, %all%                                1b3a6d
    %priorities%                                                 1b3a7
        csyprio     =    1, %system procedure%                   1b3a7a
        cdfprio     =    1, %default procedure%                  1b3a7b
    %data type limitations%                                      1b3a8
        cidxwid     =   18, %width (in bits) of INDEX%           1b3a8a
        cintwid     =   36, %width (in bits) of INTEGER%         1b3a8b
        cmxidx      =1B5-1, %max INDEX%                          1b3a8c
%Tenex user interface dependent%                                 1b3b
    %DPS operation numbers%                                      1b3b1
        civdps  =   0, %invoke system call%                      1b3b1a
        crrdps  =   1, %retrieve system call results%            1b3b1b
        cdrdps  =   2, %discard system call results%             1b3b1c
        cgtdps  =   3, %get user call arguments%                 1b3b1d
        cptdps  =   4, %return user call results%                1b3b1e
        cpgdps  =   5, %return/get user call parameters%         1b3b1f
        cerdps  =   6, %retrieve diagnostic message%             1b3b1g
        cmoprn  =   6, %max operation number%                    1b3b1h
    %VJSYS numbers%                                              1b3b2
        ccrtps  =  1B, %create remote process%                   1b3b2a
        cdelps  =  2B, %delete remote process%                   1b3b2b
        citdps  =  3B, %introduce remote processes%              1b3b2c
```

```
csepps  =  4B, %separate remote processes%                          1b3b2d

copnpk  =  5B, %open remote packages%                               1b3b2e

cclspk  =  6B, %close remote packages%                              1b3b2f

ccalpe  =  7B, %call remote procedure%                              1b3b2g

cvispe  = 10B, %visit remote callee/caller%                         1b3b2h

caloch  = 11B, %allocate remote procedure call
handle%                                                             1b3b2i

crelch  = 12B, %[abort and] release remote call
handle%                                                             1b3b2j

cacqpe  = 13B, %acquire control from remote
procedure%                                                          1b3b2k

crelpe  = 14B, %release control to remote procedure%                1b3b2l

cintpe  = 15B, %interrupt remote callee%                            1b3b2m

crsmpe  = 16B, %resume remote callee%                               1b3b2n

cntepe  = 17B, %make event known to remote caller%                  1b3b2o

chlppe  = 20B, %solicit help from remote caller%                    1b3b2p

ccrtdt  = 21B, %create remote data store%                           1b3b2q

cdeldt  = 22B, %delete remote data store%                           1b3b2r

crddt   = 23B, %read remote data store%                             1b3b2s

cwrdt   = 24B, %write remote data store%                            1b3b2t

clckdt  = 25B, %lock remote data store%                             1b3b2u

culkdt  = 26B, %unlock remote data store%                           1b3b2v

ccrtch  = 27B, %create channel between remote
processes%                                                          1b3b2w

cdelch  = 30B, %delete channel between remote
processes%                                                          1b3b2x

ccrtsp  = 31B, %create local subprocess%                            1b3b2y
```

```
cdelsp  = 32B, %delete local subprocess%                1b3b2z
ccrtpr  = 33B, %create local processor%                 1b3b2a@
cdelpr  = 34B, %delete local processor%                 1b3b2aa
csipr   = 35B, %sign in local processor%                1b3b2ab
crdypr  = 36B, %ready local processor%                  1b3b2ac
csndch  = 37B, %send data on local channel%             1b3b2ad
crcvch  = 40B, %receive data on local channel%          1b3b2ae
ccrtlk  = 41B, %create local lock%                      1b3b2af
cdellk  = 42B, %delete local lock%                      1b3b2ag
csetlk  = 43B, %set local lock%                         1b3b2ah
cremlk  = 44B, %unset local lock%                       1b3b2ai
ccrtev  = 45B, %create local event%                     1b3b2aj
cdelev  = 46B, %delete local event%                     1b3b2ak
csigev  = 47B, %signal local event%                     1b3b2al
ctstev  = 50B, %test for signalled local event%         1b3b2am
cwaiev  = 51B, %wait for local event to be signalled%
                                                        1b3b2an
cinfps  = 52B, %retrieve info about remote process%     1b3b2ao
csopr   = 53B, %signout local processor%                1b3b2ap
citdfk  = 54B, %introduce fork to DPS%                  1b3b2aq
csepfk  = 55B, %separate fork from DPS%                 1b3b2ar
csigpe  = 56B, %signal remote caller/callee%            1b3b2as
csetmr  = 57B, %set interval timer%                     1b3b2at
ctstmr  = 60B, %test/cancel interval timer%             1b3b2au
csetrc  = 61B, %set remote DPS trace word%              1b3b2av
cmsycn  = 61B, %max system call number%                 1b3b2aw
```

```
%VJUSR numbers%                                                    1b3b3
   cprso   =  1B, %solicits sign out of local processor%           1b3b3a
   cinipk  =  2B, %initialize local package%                       1b3b3b
   ctrmpk  =  3B, %terminate local package%                        1b3b3c
   cpecal  =  4B, %call local procedure%                           1b3b3d
   cpeint  =  5B, %interrupt local procedure%                      1b3b3e
   cpersm  =  6B, %resume local procedure%                         1b3b3f
   cpeabr  =  7B, %abort local procedure%                          1b3b3g
   cpente  = 10B, %make event known to local caller%               1b3b3h
   cpehlp  = 11B, %solicit help from local caller%                 1b3b3i
   clvrdt  = 12B, %verify local data store's existence%            1b3b3j
   clrddt  = 13B, %read local data store%                          1b3b3k
   clwrdt  = 14B, %write local data store%                         1b3b3l
   cokips  = 15B, %OK introduction to local process%               1b3b3m
   coksps  = 16B, %OK separation from local process%               1b3b3n
   cokopk  = 17B, %OK opening of local package%                    1b3b3o
   cokcpk  = 20B, %OK closing of local package%                    1b3b3p
   cokcch  = 21B, %OK creation of channel to local
   process%                                                        1b3b3q
   cokdch  = 22B, %OK deletion of channel to local
   process%                                                        1b3b3r
   cntlch  = 23B, %note loss of channel to remote
   process%                                                        1b3b3s
   coklps  = 24B, %OK splicing of remote process%                  1b3b3t
   cokups  = 25B, %OK unsplicing of remote process%                1b3b3u
   cmuscn  = 25B, %max user call number%                           1b3b3v
```

```
        %miscellaneous%                                                 1b3b4

            chostaddr=   1, %host address process info type%            1b3b4a

            cjuevlen =  16, %length of JUSR event%                      1b3b4b

            cmnnpr   =   1, %min number processors/subprocess%          1b3b4c

            cmxnpr   =   4, %max number processors/subprocess%          1b3b4d

            cmevlen  =  16, %max length of event%                       1b3b4e

            cmecbs   =  20, %max length of ECB list%                    1b3b4f

            cmsycargl=   8, %max # system call args
            (SYSCAL/SYSBGN)%                                            1b3b4g

            cmsycresl=   3, %max # system call ress
            (SYSCAL/SYSEND)%                                            1b3b4h

            cmuscargl=   7, %max # user call args (VJUSR)%              1b3b4i

            cmuscresl=   3, %max # user call ress (VJUSR)%              1b3b4j

    %network interface dependent%                                       1b3c

        %message numbers%                                               1b3c1

            cmcalpe  =   1, %call local procedure%                      1b3c1a

            cmrecpe  =   2, %recall local procedure%                    1b3c1b

            cmrtnpe  =   3, %return to local caller%                    1b3c1c

            cmerror  =   4, %error%                                     1b3c1d

            cmmsgn   =   4, %max message number%                        1b3c1e

        %system procedure numbers%                                      1b3c2

            cxinips  =   1, %initialize process%                        1b3c2a

            cxtrmps  =   2, %terminate process%                         1b3c2b

            cxcrtchh =   3, %create channel half%                       1b3c2c

            cxdelchh =   4, %delete channel half%                       1b3c2d

            cxopnpk  =   5, %open packages%                             1b3c2e
```

```
        cxclspk  =   6, %close packages%                          1b3c2f
        cxintpe  =   7, %interrupt procedure%                     1b3c2g
        cxrsmpe  =   8, %resume procedure%                        1b3c2h
        cxabrpe  =   9, %abort procedure%                         1b3c2i
        cxhlppe  =  10, %solicit help with procedure%             1b3c2j
        cxcrtdt  =  11, %create data store%                       1b3c2k
        cxdeldt  =  12, %delete data store%                       1b3c2l
        cxrddt   =  13, %read data store%                         1b3c2m
        cxwrdt   =  14, %write data store%                        1b3c2n
        cxlckdt  =  15, %lock data store%                         1b3c2o
        cxulkdt  =  16, %unlock data store%                       1b3c2p
        cxrdmnu  =  17, %read channel type menu%                  1b3c2q
        cxcrtce  =  18, %create channel end%                      1b3c2r
        cxalopo  =  19, %allocate port%                           1b3c2s
        cxrelpo  =  20, %release port%                            1b3c2t
        cxntepe  =  21, %make event known to local caller%        1b3c2u
        cxsigpe  =  22, %signal remote caller/callee%             1b3c2v
        cxsetrc  =  23, %set DPS trace word%                      1b3c2w
        cmsypn   =  23, %max system procedure number%             1b3c2x
    %message parameters%                                          1b3c3
        cmrout   =   1, %msg routing code elem number%            1b3c3a
        cmnumb   =   2, %msg number elem number%                  1b3c3b
        cmhdrc   =   2, %length of message header%                1b3c3c
        cmmsgl   =   9, %max length of message%                   1b3c3d
        cmpdcl   =   4, %length of process description%           1b3c3e
```

```
        cmplocl  =   3, %length of process location%                    1b3c3f

        cmrpeargl=   7, %max # remote proc args
        (SYSCAL/SYSBGN)%                                                1b3c3g

        cmrperesl=   3, %max # remote proc ress
        (SYSCAL/SYSEND)%                                                1b3c3h

    %implementation dependent%                                          1b3d

      %trace bits%                                                      1b3d1

        ctrimsg    = 4B11, %incoming messages%                          1b3d1a

        ctromsg    = 2B11, %outgoing messages%                          1b3d1b

        ctrfmsg    = 1B11, %self messages%                              1b3d1c

        ctrsarg    = 4B10, %system call arguments%                      1b3d1d

        ctrsres    = 2B10, %system call results%                        1b3d1e

        ctrsabr    = 1B10, %system call abort%                          1b3d1f

        ctruarg    =  4B9, %user call arguments%                        1b3d1g

        ctrures    =  2B9, %user call results%                          1b3d1h

        ctruabr    =  1B9, %user call abort%                            1b3d1i

        ctrlkwt    =  4B8, %manager held/released for lock%             1b3d1j

        ctrerrs    =  2B8, %errors%                                     1b3d1k

      %NOTE values%                                                     1b3d2

        cselpr     =    1, %processor selected%                         1b3d2a

        crelpr     =    2, %release processor%                          1b3d2b

      %IPC routine numbers%                                             1b3d3

        cicrtps    =    1, %create process%                             1b3d3a

        cidelps    =    2, %delete process%                             1b3d3b

        cicrtce    =    3, %create end of physical channel%             1b3d3c

        cidelce    =    4, %delete end of physical channel%             1b3d3d
```

```
        cialopo    =    5, %allocate port%                    1b3d3e
        cirelpo    =    6, %release port%                     1b3d3f
        cisndch    =    7, %send data to process%             1b3d3g
        circvch    =    8, %receive data from process%        1b3d3h
        ciupdpo    =    9, %update port states%               1b3d3i
        cmipcn     =    9, %max IPC routine number%           1b3d3j
    %process types%                                           1b3d4
        cself      =    1, %self%                             1b3d4a
        csuperior  =    2, %direct superior%                  1b3d4b
        cinferior  =    3, %direct inferior%                  1b3d4c
        cintroduced=    4, %introduced%                       1b3d4d
    %specialized lock types%                                  1b3d5
        ccreate    =    3, %create%                           1b3d5a
        cdelete    =    4, %delete%                           1b3d5b
        csend      =    5, %send message%                     1b3d5c
        creceive   =    6, %receive message%                  1b3d5d
        cinternal  =    7, %internal%                         1b3d5e
        cexternal  =    8, %external%                         1b3d5f
        csignal    =    9, %signal event%                     1b3d5g
        cwait      =   10, %await event%                      1b3d5h
        copncls    =   11, %open/close%                       1b3d5i
    %parameter list types%                                    1b3d6
        carglist   =    1, %argument list%                    1b3d6a
        creslist   =    2, %result list%                      1b3d6b
        cabrlist   =    3, %abort list%                       1b3d6c
```

```
%call states%                                                    1b3d7
   cuncalled  =    1, %uncalled%                                 1b3d7a
   ccompleted =    2, %completed%                                1b3d7b
   cinterrupted=   3, %interrupted%                              1b3d7c
   creturned  =    4, %returned%                                 1b3d7d
   crunloc    =    5, %running locally%                          1b3d7e
   crunrem    =    6, %running remotely%                         1b3d7f
   cstuck     =    7, %stuck%                                    1b3d7g
%user call states%                                               1b3d8
   %cuncalled =    1,  uncalled%                                 1b3d8a
   %ccompleted=    2,  completed%                                1b3d8b
   crunning   =    3, %running%                                  1b3d8c
%record attributes%                                              1b3d9
   cph        =    1, %process handle%                           1b3d9a
   cpcrh      =    2, %processor handle%                         1b3d9b
%process address actions%                                        1b3d10
   csplice    =    2, %splice to old process%                    1b3d10a
   %ccreate   =    3,  create new process%                       1b3d10b
%entity types%                                                   1b3d11
   cblock     =    1, %block%                                    1b3d11a
   %ccharstr  =    6,  character string%                         1b3d11b
   %clist     =    7,  list%                                     1b3d11c
   %cstruc    =    8,  B36 data structure%                       1b3d11d
%RUNFLD operations%                                              1b3d12
   cprevue    =    1, %prevue%                                   1b3d12a
```

```
            clock       =    2, %lock%                              1b3d12b
            cuse        =    3, %use%                               1b3d12c
            crewind     =    4, %rewind%                            1b3d12d
         %UNLCKREC dispositions%                                     1b3d13
            ckeep       =    1, %keep%                              1b3d13a
            %cdelete    =    4,  delete%                            1b3d13b
            cdspmsk     =    5, %,X inversion mask%                 1b3d13c
         %call types%                                                1b3d14
            clocal      =    1, %local%                             1b3d14a
            cremote     =    2, %remote%                            1b3d14b
         %manager trace types%                                       1b3d15
            chold       =    1, %held%                              1b3d15a
            crelease    =    2, %released%                          1b3d15b
         %miscellaneous%                                             1b3d16
            cimmprio    =  1B5, %immediate priority%                1b3d16a
            cmsmlblk    =    3, %max length of "small" block%       1b3d16b
            cmcmnul     =    3, %max length of channel menu%        1b3d16c
            cmargl      =   20, %max # args (L10CAL)%               1b3d16d
            cmresl      =   20; %max # ress (L10CAL)%               1b3d16e
   %local error numbers% DECLARE CONSTANT                            1b4
      %character 1 legend%                                           1b4a
         % e -- error %                                             1b4a1
      %character 2 legend%                                           1b4b
         % c -- inter-process communication %                       1b4b1
         % d -- data structure %                                    1b4b2
```

```
        % e -- error %                                              1b4b3
        % f -- folder %                                             1b4b4
        % g -- undefined handle%                                    1b4b5
        % h -- no available handle%                                 1b4b6
        % k -- package %                                            1b4b7
        % l -- lock %                                               1b4b8
        % m -- storage %                                            1b4b9
        % p -- procedure %                                          1b4b10
        % r -- processor %                                          1b4b11
        % s -- process %                                            1b4b12
        % v -- event %                                              1b4b13
      %character 3 legend%                                         1b4c
        % d -- duplicate %                                         1b4c1
        % f -- failed %                                            1b4c2
        % i -- illegal %                                           1b4c3
        % m -- missing %                                           1b4c4
        % o -- overflow %                                          1b4c5
        % u -- undefined %                                         1b4c6
        % w -- wrong %                                             1b4c7
      %------------------%                                         1b4d
      %data structures%                                            1b4e
        edddst =   1, %destination supplied for decoded BITSTR%   1b4e1
        eddkey =   2, %structure already has key%                  1b4e2
        eddrky =   3, %key has key%                                1b4e3
        eddsto =   4, %duplicate data store%                       1b4e4
```

```
        ediidx =   5, %illegal index%                          1b4e5
        edipdl =   6, %illegal psel/dsel%                      1b4e6
        ediuif =   7, %illegal user information%               1b4e7
        edmkey =   8, %no key%                                 1b4e8
        edodcl =   9, %dst too small to decode list%           1b4e9
        edolst =  10, %max list size exceeded%                 1b4e10
        edostr =  11, %dst too small to decode string%         1b4e11
        edufty =  12, %undefined formal data type%             1b4e12
        eduity =  13, %undefined informal data type%           1b4e13
        edusto =  14, %undefined data store name%              1b4e14
        edwesl =  15, %element selector applied to non-list%   1b4e15
        edwidx =  16, %no such index%                          1b4e16
        edwkey =  17, %no such key%                            1b4e17
        edwtyp =  18, %wrong data type%                        1b4e18
        edwpmc =  19, %incorrect number of parameters%         1b4e19
        edoabc =  20, %data structure overflows source ABC%    1b4e20
    %errors%                                                   1b4f
        eefimp =  51, %not implemented%                        1b4f1
        eefops =  52, %operating system error%                 1b4f2
        eemerr =  53, %unidentifiable operating system error%  1b4f3
        eeuern =  54, %undefined error number%                 1b4f4
        eefl10 =  55, %L10 run-time error%                     1b4f5
    %events%                                                   1b4g
        evfacq = 101, %won't delete acquisition event%         1b4g1
        evfjuv = 102, %won't delete JUSR event%                1b4g2
```

```
        evilen = 103, %illegal event length%                    1b4g3
        evoecb = 104, %ECB overflow%                            1b4g4
        evftmr = 105, %won't delete timer event%                1b4g5
    %folders%                                                   1b4h
        efodrn = 151, %create record while drained%             1b4h1
        eforun = 152, %RUNFLD overrun%                          1b4h2
        efuift = 153, %undefined record information type%       1b4h3
        efurop = 154, %undefined RUNFLD operation%              1b4h4
    %inter-process communication%                               1b4i
        ecdcce = 201, %duplicate create channel on port%        1b4i1
        ecutyp = 202, %undefined channel type%                  1b4i2
        ecwmnu = 203, %channel type menu mismatch%              1b4i3
        ecwpkl = 204, %inconsistent packet length%              1b4i4
    %locks%                                                     1b4j
        elfded = 251, %deadlock%                                1b4j1
        elfdel = 252, %sought lock deleted%                     1b4j2
        elflck = 253, %lock attempt failed%                     1b4j3
        elmswp = 254, %non-existent LCB to be swapped%          1b4j4
        elmstk = 255, %lock stack underflow%                    1b4j5
        elostk = 256, %lock stack overflow%                     1b4j6
        elistk = 257, %lock stack surplus%                      1b4j7
    %packages%                                                  1b4k
        ekfded = 301, %package dead%                            1b4k1
        ekupkn = 302, %no such package%                         1b4k2
    %procedures%                                                1b4l
```

```
epfnoh = 351, %no help available%                                   1b411
epfpio = 352, %no processor with sufficient priority%               1b412
epfsab = 353, %won't abort system procedure%                        1b413
epfsin = 354, %won't interrupt system procedure%                    1b414
epiacq = 355, %wrong state for ACQPE%                               1b415
epiaid = 356, %action on behalf of "idle" local
procedure%                                                          1b416
epihlp = 357, %wrong state for HLPPE%                               1b417
epiint = 358, %wrong state for INTPE%                               1b418
epimsk = 359, %illegal argument/result list mask parm%              1b419
epinte = 360, %wrong state for NTEPE%                               1b4110
epiotc = 361, %illegal system procedure outcome%                    1b4111
epirel = 362, %wrong state for RELPE%                               1b4112
epirsm = 363, %wrong state for RSMPE%                               1b4113
epixhp = 364, %wrong state for call to XHLPPE%                      1b4114
epixin = 365, %wrong state for call to XINTPE%                      1b4115
epixnt = 366, %wrong state for sending XNTEPE%                      1b4116
epixrc = 367, %wrong state for sending XRECPE%                      1b4117
epixrm = 368, %wrong state for call to XRSMPE%                      1b4118
epixrn = 369, %wrong state for sending XRTNPE%                      1b4119
epuotc = 370, %undefined procedure outcome%                         1b4120
epurtn = 371, %undefined return type%                               1b4121
epusyn = 372, %undefined system procedure number%                   1b4122
epwvis = 373, %unplanned for visit%                                 1b4123
epfqin = 374, %won't interrupt sequential processor%                1b4124
```

```
        episig = 375, %wrong state for SIGPE%                    1b4125
        epixsg = 376, %wrong state for XSIGPE%                   1b4126
    %processes%                                                  1b4m
        esdpoh = 401, %POH already associated with PH%           1b4m1
        esfded = 402, %process dead%                             1b4m2
        esipsa = 403, %syntax error in process addr%             1b4m3
        esircv = 404, %RCVPS to process with no POH%             1b4m4
        esisup = 405, %not direct superior%                      1b4m5
        esmpoh = 406, %no POH associated with PH%                1b4m6
        esumsg = 407, %undefined message number%                 1b4m7
        esuser = 408, %undefined user name%                      1b4m8
        esuwtd = 409, %undefinedwatchdog code%                   1b4m9
        eswqak = 410, %no process query acknowledgment%          1b4m10
    %processors%                                                 1b4n
        erigtd = 451, %wrong state for GTDPS%                    1b4n1
        eriptd = 452, %wrong state for PTDPS%                    1b4n2
        erisin = 453, %not signed in%                            1b4n3
        erorsb = 454, %resource block overflow%                  1b4n4
        erosml = 455, %small block overflow%                     1b4n5
        erowin = 456, %processor window overflow%                1b4n6
        erualo = 457, %allocate undefined entity type%           1b4n7
        eruinf = 458, %undefined process info type%              1b4n8
        eruopn = 459, %undefined operation number%               1b4n9
        erupml = 460, %undefined parameter location%             1b4n10
        erupsi = 461, %undefined PSI channel number%             1b4n11
```

```
        erurde = 462, %read undefined entity type%                      1b4n12
        eruscp = 463, %undefined scope%                                 1b4n13
        erusyc = 464, %undefined system call number%                    1b4n14
        eruusc = 465, %undefined user call number%                      1b4n15
        eruwre = 466, %write undefined entity type%                     1b4n16
        erfded = 467, %processor dead%                                  1b4n17
        erdsin = 468, %duplicate signin%                                1b4n18
    %storage%                                                           1b4o
        emfexh = 501, %CF storage exhausted%                            1b4o1
        emient = 502, %zero/negative entity size%                       1b4o2
        emuent = 503, %undefined entity type%                           1b4o3
        emibyp = 504, %illegal user-supplied byte pointer%              1b4o4
        emiadr = 505, %illegal user-supplied address%                   1b4o5
        emxern = 615; %max error number (see also DPSTBL)%              1b4o6
%global record definitions%                                             1b5
    (car) RECORD %call record%                                          1b5a
        cacost    [cintwid], %cumulative cost of call%                  1b5a1
        cacch     [cidxwid], %controlling/chaining call handle%         1b5a2
        caprio    [cidxwid], %priority%                                 1b5a3
        capkh     [cidxwid], %package handle%                           1b5a4
        capname   [ADDRESS], %addr of procedure name/number
        (B36)%                                                          1b5a5
        caentry   [ADDRESS], %addr of system procedure table
        entry%                                                          1b5a6
        caprml    [ADDRESS], %addr of parameter list%                   1b5a7
        camskl    [ADDRESS], %addr of parameter list mask%              1b5a8
```

```
    caqecb    [ADDRESS], %addr of acquisition ECB%                1b5a9
    carecb    [ADDRESS], %addr of acquisition ack ECB%            1b5a10
    caph      [chndwid], %process handle%                         1b5a11
    casph     [chndwid], %subprocess handle%                      1b5a12
    capcrh    [chndwid], %processor handle%                       1b5a13
    caqevh    [chndwid], %acquisition event handle%               1b5a14
    castate         [3], %state%                                  1b5a15
    caprvstate      [3], %pre-interruption state%                 1b5a16
    caseqpr         [1], %sequential processor%                   1b5a17
    caremote        [1]; %remote call%                            1b5a18
%+PDP10% (cnr) RECORD %physical channel record%                   1b5b
    cnpoh1    [cidxwid], %port handle 1%                          1b5b1
    cnpoh2    [cidxwid], %port handle 2%                          1b5b2
    cnph1     [chndwid], %process handle 1%                       1b5b3
    cnph2     [chndwid], %process handle 2%                       1b5b4
    cn1dead         [1], %side 1 dead%                            1b5b5
    cn2dead         [1]; %side 2 dead%%+PDP10%                    1b5b6
(defr) RECORD %folder definition record%                          1b5c
    dfflda      [WORD], %addr for folder addr%                    1b5c1
    dfrcdf      [WORD], %addr of record definition%               1b5c2
    dfinit      [WORD], %addr of record initializer%              1b5c3
    dfterm      [WORD], %addr of record terminator%               1b5c4
    dfhand      [WORD], %addr of handle absoluter%                1b5c5
    dfrcwl    [chndwid], %length of record workspace%             1b5c6
    dfmnhd    [chndwid], %min handle%                             1b5c7
```

```
    dfmxhd    [chndwid], %max handle%                                  1b5c8
    dfpad     [ADDRESS], %padding%                                     1b5c9
    dfennex         [12], %error number for non-existent
    handle%                                                            1b5c10
    dfenovf         [12], %error number for folder overflow%           1b5c11
    dfmapl           [2]; %length (in words) of handle map%            1b5c12
(dsr) RECORD %B36 data structure%                                      1b5d
    dslen           [15], %length%                                     1b5d1
    ds1unused        [3], %unused%                                     1b5d2
    dstype           [4], %type%                                       1b5d3
    ds2unused       [13], %unused%                                     1b5d4
    dskey            [1]; %key%                                        1b5d5
%+PDP10% (dtr) RECORD %data store record%                              1b5e
    dtname    [ADDRESS], %addr of UC name%                             1b5e1
    dthash    [ADDRESS], %name hash%                                   1b5e2
    dtvalue   [ADDRESS], %addr of value%                               1b5e3
    dtlcb     [ADDRESS], %addr of LCB%                                 1b5e4
    dtpkh     [chndwid], %package handle%                              1b5e5
    dtuse     [chndwid], %lock use count%                              1b5e6
    dtdummy          [1]; %for locking purposes only%%+PDP10%          1b5e7
(enr) RECORD %table entry header%                                      1b5f
    enpadr       [WORD], %addr of procedure%                           1b5f1
    enresc           [9], %number of results%                          1b5f2
    enargc           [9]; %number of arguments%                        1b5f3
%+PDP10% (evr) RECORD %event record%                                   1b5g
```

```
    evecb       [ADDRESS], %addr of ECB%                          1b5g1
    evchan            [9]; %PSI channel number%%+PDP10%           1b5g2
(fldr) RECORD %folder record%                                     1b5h
    fdchnf      [ADDRESS], %addr of first record%                 1b5h1
    fdchnb      [ADDRESS], %addr of last record%                  1b5h2
      %maintain position of above fields%                         1b5h2a
    fddef       [ADDRESS], %addr of folder definition%            1b5h3
    fdreccnt    [chndwid], %record count%                         1b5h4
    fduse       [chndwid], %RUNFLD use count%                     1b5h5
    fddrain           [1]; %creation of new records disabled%     1b5h6
(lcbr) RECORD %lock control block%                                1b5i
    lcpcrid     [ADDRESS], %addr of immediate processor id%       1b5i1
    lcimsh            [9], %no. share-locks in LCB%               1b5i2
    lcimex            [9], %no. exclusive-locks in LCB%           1b5i3
    lcfdsh            [9], %no. share-locks in folder%            1b5i4
    lcfdex            [9], %no. exclusive-locks in folder%        1b5i5
    lcfdsp            [9], %no. specialized-locks in folder%      1b5i6
    lcfdrq            [9]; %no. lock requests in folder%          1b5i7
(ldscr) RECORD %lockset descriptor%                               1b5j
    ldlcb       [ADDRESS], %addr of LCB%                          1b5j1
    ldlsh       [chndwid], %lockset handle%                       1b5j2
    ldtype            [3], %type%                                 1b5j3
    lddead            [1]; %dead%                                 1b5j4
(lkr) RECORD %lock record%                                        1b5k
    lklcb       [ADDRESS]; %addr of LCB%                          1b5k1
```

```
(lsr) RECORD %lock set record%                                  1b51
   lslckid   [ADDRESS], %addr of lock id%                       1b511
   lspcrid   [ADDRESS], %addr of processor id%                  1b512
   lsecb     [ADDRESS], %addr of ECB%                           1b513
   lslocked        [1], %locked%                                1b514
   lshandle        [1]; %handle specifically requested%         1b515
(mgr) RECORD %manager record%                                   1b5m
   mgstup       [WORD], %startup parameter%                     1b5m1
   mgroot    [ADDRESS], %addr of top procedure%                 1b5m2
   mgaddr    [ADDRESS], %addr of top procedure%                 1b5m3
   mgecbs    [ADDRESS], %addr of ECB [addr list]%               1b5m4
   mgacs     [ADDRESS], %addr of saved ACs%                     1b5m5
   mgstate   [ADDRESS], %addr of saved DPS state%               1b5m6
   mgfwbase        [9], %state file base page number%           1b5m7
   mgrscnt   [chndwid], %restart ceiling%                       1b5m8
   mgurscnt  [chndwid], %restart count%                         1b5m9
   mglon           [1]; %ECB list%                              1b5m10
(pkr) RECORD %package record%                                   1b5n
   pkcost    [cintwid], %cumulative cost of package%            1b5n1
   pkintpkh  [chndwid], %internal package handle%               1b5n2
   pksph     [chndwid], %subprocess handle%                     1b5n3
   pkdead          [1]; %dead%                                  1b5n4
(por) RECORD %port record%                                      1b5o
   %common%                                                     1b5o1
      posndecb [ADDRESS], %addr of send ECB%                    1b5o1a
```

```
        porcvecb [ADDRESS], %addr of receive ECB%                    1b5o1b
        potype       [2], %port type%                                1b5o1c
        poformat     [2], %port format%                              1b5o1d
        poactive     [1], %active end of channel%                    1b5o1e
        pocreated    [1], %channel created%                          1b5o1f
    %inter-fork dependent%                                           1b5o2
        posndwd  [ADDRESS], %addr of send half window%               1b5o2a
        porcvwd  [ADDRESS], %addr of receive half window%            1b5o2b
        pofkh    [ADDRESS], %fork handle%                            1b5o2c
        polwbase     [9], %local window base page number%            1b5o2d
        porwbase     [9]; %remote window base page number%           1b5o2e
(prr) RECORD %processor record%                                      1b5p
    prcost    [cintwid], %cumulative cost of processor%              1b5p1
    prfkh     [ADDRESS], %fork handle%                               1b5p2
    prstup    [ADDRESS], %addr of startup info%                      1b5p3
    prsiecb   [ADDRESS], %addr of signin ECB%                        1b5p4
    prrbaddr  [ADDRESS], %addr of resource block%                    1b5p5
    prrbm     [ADDRESS], %max resource block length%                 1b5p6
    prrbl     [ADDRESS], %cur resource block length%                 1b5p7
    prprio    [cidxwid], %priority%                                  1b5p8
    prsph     [chndwid], %subprocess handle%                         1b5p9
    prsuper   [chndwid], %supervisor's processor handle%             1b5p10
    prjuevh   [chndwid], %JUSR event handle%                         1b5p11
    prrycnt   [chndwid], %ready count%                               1b5p12
    prush     [chndwid], %current user call handle%                  1b5p13
```

```
        prabrchn        [6], %abort PSI channel%                        1b5p14
        prspldr         [1], %subprocess leader%                        1b5p15
        prsecond        [1], %secondary processor%                      1b5p16
        prsignin        [1], %signed in%                                1b5p17
        prautordy       [1], %system to ready processor%                1b5p18
        prseqpr         [1], %sequential processor%                     1b5p19
        prrunning       [1], %running%                                  1b5p20
        prdead          [1], %dead%                                     1b5p21
        prrbovflw       [1]; %resource block overflow%                  1b5p22
    (psr) RECORD %process record%                                       1b5q
        pscost    [cintwid], %cumulative cost of process%               1b5q1
        pspsdesc  [ADDRESS], %addr of process description (B36)%        1b5q2
        psadjsh   [cidxwid], %adjacent segment handle%                  1b5q3
        psadjph   [chndwid], %adjacent process handle%                  1b5q4
        pspoh     [chndwid], %port handle%                              1b5q5
        pstype          [3], %process type%                             1b5q6
        psdead          [1]; %dead%                                     1b5q7
    (recr) RECORD %record record%                                       1b5r
        rcchnf    [ADDRESS], %addr of next record%                      1b5r1
        rcchnb    [ADDRESS], %addr of previous record%                  1b5r2
           %maintain position of above fields%                          1b5r2a
        rcfld     [ADDRESS], %addr of folder%                           1b5r3
        rcpcrid   [ADDRESS], %addr of processor id%                     1b5r4
        rclckid   [ADDRESS], %addr of lock id%                          1b5r5
        rclcb     [ADDRESS], %addr of LCB%                              1b5r6
```

```
    rchnd       [chndwid], %record handle%                          1b5r7
    rcdeleted         [1]; %deleted%                                1b5r8
      DECLARE CONSTANT crcost =                                     1b5r8a
        recr,SIZE+cmpidl+cmlkidl+lcbr,SIZE+cmpidl;                  1b5r8a1
  (sgr) RECORD %segment record%                                     1b5s
    sgsh1       [cidxwid], %segment handle 1%                       1b5s1
    sgsh2       [cidxwid], %segment handle 2%                       1b5s2
    sgph1       [chndwid], %process handle 1%                       1b5s3
    sgph2       [chndwid], %process handle 2%                       1b5s4
    sgpch       [chndwid], %physical channel handle%                1b5s5
    sg1dead           [1], %side 1 dead%                            1b5s6
    sg2dead           [1], %side 2 dead%                            1b5s7
    sgintro           [1]; %introduction%                           1b5s8
  (specr) RECORD %processor parameter spec%                         1b5t
    spdtype           [7], %type%                                   1b5t1
    sploc             [2], %intra-AC location%                      1b5t2
    spac              [2], %AC number - 1%                          1b5t3
    sppad             [1]; %important! padding%                     1b5t4
  (sur) RECORD %subprocess record%                                  1b5u
    sucost      [cintwid], %cost of subprocess%                     1b5u1
    susuaddr    [ADDRESS], %addr of subprocess address%             1b5u2
    supknames   [ADDRESS], %addr of package name list%              1b5u3
    supkhashs   [ADDRESS], %addr of package name hash list%         1b5u4
    supkuses    [ADDRESS], %addr of package use count list%         1b5u5
    suwtecbs    [ADDRESS], %addr of ECB of waiting managers%        1b5u6
```

```
    sufkh      [ADDRESS], %subprocess leader's fork handle%      1b5u7
    suefkh     [ADDRESS], %encapsulator's fork handle%           1b5u8
    suepcrh    [chndwid], %encapsulator's processor handle%      1b5u9
    supcrh     [chndwid], %subprocess leader's handle%           1b5u10
    sucrnpr    [chndwid], %cur processors count%                 1b5u11
    sumnnpr    [chndwid], %min processors count%                 1b5u12
    sumxnpr    [chndwid], %max processors count%                 1b5u13
    supg             [9], %shared segment base page number%      1b5u14
    supgcnt          [9], %shared segment page count%            1b5u15
    supsldr          [1], %process leader%                       1b5u16
    susignin         [1], %signed in%                            1b5u17
    suautopcr        [1]; %system to demand-create processors%   1b5u18
(syr) RECORD %system call record%                                1b5v
    syentry    [ADDRESS], %addr of table entry%                  1b5v1
    syargl     [ADDRESS], %addr of argument list%                1b5v2
    syresl     [ADDRESS], %addr of result list%                  1b5v3
    synumb     [chndwid], %system call number%                   1b5v4
    syabort          [1]; %aborted%                              1b5v5
(tmr) RECORD %timer record%                                      1b5w
    tminterval   [WORD], %initial interval in ms%                1b5w1
    tmleft       [WORD], %remaining interval in ms%              1b5w2
    tmecb     [ADDRESS], %addr of ECB%                           1b5w3
    tmevh      [chndwid]; %event handle%                         1b5w4
(typer) RECORD %data type%                                       1b5x
    tyfunt           [5], %fundamental type%                     1b5x1
```

```
        tyempok         [1], %EMPTY OK in lieu of fund type%          1b5x2
        tylist          [1]; %list of fund type structures%           1b5x3
    (usr) RECORD %user call record%                                   1b5y
        uscost      [cintwid], %cost of user call%                    1b5y1
        usentry     [ADDRESS], %addr of table entry%                  1b5y2
        usecb       [ADDRESS], %addr of ECB%                          1b5y3
        usargl      [ADDRESS], %addr of argument list%                1b5y4
        usresl      [ADDRESS], %addr of result list%                  1b5y5
        usdgmsg     [ADDRESS], %addr of diagnostic message%           1b5y6
        usnumber    [chndwid], %number%                               1b5y7
        uscph       [chndwid], %controlling process handle%           1b5y8
        usspH       [chndwid], %subprocess handle%                    1b5y9
        uspcrh      [chndwid], %processor handle%                     1b5y10
        usstate         [2]; %state%                                  1b5y11
%global catchphrases%                                                 1b6

    (palwkp)  CATCHPHRASE; unlckrec (CATCHPARAM, always
    (ckeep));                                                         1b6a

    (palwdl)  CATCHPHRASE; unlckrec (CATCHPARAM, always
    (cdelete));                                                       1b6b

    (prtnkp)  CATCHPHRASE; unlckrec (CATCHPARAM, ifrtn
    (ckeep));                                                         1b6c

    (prtndl)  CATCHPHRASE; unlckrec (CATCHPARAM, ifrtn
    (cdelete));                                                       1b6d

    (prstwhl) CATCHPHRASE; IF abrtn () THEN vwheel _ CATCHPARAM;      1b6e

    (ppoplcb) CATCHPHRASE; IF abrtn () THEN poplcb ();                1b6f

    (pignore) CATCHPHRASE; IF abr    () THEN TERMINATE;               1b6g

    (palwrb)  CATCHPHRASE;                                            1b6h
```

```
      IF abrtn () THEN relent (cblock, CATCHPARAM);                    1b6h1
   (palwrl)  CATCHPHRASE;                                              1b6i
      IF abrtn () THEN relent (clist,  CATCHPARAM);                    1b6i1
   (pabrrb)  CATCHPHRASE;                                              1b6j
      IF abr   () THEN relent (cblock, CATCHPARAM);                    1b6j1
   (pabrrl)  CATCHPHRASE;                                              1b6k
      IF abr   () THEN relent (clist,  CATCHPARAM);                    1b6k1
   (pignuch) CATCHPHRASE;                                              1b6l
      IF abr   () AND SIGNAL = egmhca THEN TERMINATE;                  1b6l1
   (prstctx) CATCHPHRASE; IF abrtn () THEN                             1b6m
      oblkxfr (CATCHPARAM, $vctx, cmctxl);                             1b6m1
   (pfail)   CATCHPHRASE; IF abr   () THEN                             1b6n
      BEGIN                                                            1b6n1
      [CATCHPARAM] _ FALSE;                                            1b6n2
      TERMINATE;                                                       1b6n3
      END;                                                             1b6n4
   (pcover)  CATCHPHRASE (; vdgmsg); IF abr () THEN                    1b6o
      BEGIN                                                            1b6o1
      oprtstr (vdgmsg);                                                1b6o2
      LOOP oendfk (0);                                                 1b6o3
      END;                                                             1b6o4
   (pnotmgr) CATCHPHRASE (: vdgmsg);                                   1b6p
      IF abr () AND SIGNAL # erorsb THEN                               1b6p1
         BEGIN                                                         1b6p1a
         [CATCHPARAM].usdgmsg _ savent (ccharstr, vdgmsg);             1b6p1b
```

```
            sigecb ([CATCHPARAM].usecb, SIGNAL);                    1b6p1c
            END;                                                    1b6p1d
        (puldldt) CATCHPHRASE;                                      1b6q
          BEGIN                                                     1b6q1
          IF abr () THEN BUMP DOWN [CATCHPARAM].dtuse;              1b6q2
          unlckrec (CATCHPARAM,                                     1b6q3
            always (IF [CATCHPARAM].dtuse THEN ckeep ELSE
            cdelete));                                              1b6q3a
          END;                                                      1b6q4
%main program%                                                      1c
    (dpsmain) %main program%
    PROCEDURE;                                                      1c1
      %declarations%                                                1c1a
        LOCAL i, key, type, chntype, poh;                           1c1a1
        LOCAL STRING intrahostaddr [cmiahal];                       1c1a2
        LOCAL LIST menu [cmcmnul];                                  1c1a3
        LOCAL nxtdef POINTER, ps POINTER, po POINTER;               1c1a4
      %intercept terminal aborts%                                   1c1b
        INVOKE (pcover);                                            1c1b1
      %clear DPS variables to zero%                                 1c1c
        variables _ 0;                                              1c1c1
        oblkxfr ($variables, $variables+1, $varend-$variables-1);   1c1c2
      %initialize DPS state%                                        1c1d
        verrmsg,M _ cmerml;                                         1c1d1
        vpsname,M _ cmpsnl;                                         1c1d2
        vwtdecb,M _ cmwtdel;                                        1c1d3
```

```
%initialize operating system%                                    1c1e
   oinios ();                                                    1c1e1
%initialize IPC%                                                 1c1f
   vwnmp _ -1;                                                   1c1f1
   oblkxfr ($vwnmp, $vwnmp+1, cwnmpl-1);                         1c1f2
   vwinbas _ encstruc (cindex, cppown);                          1c1f3
   key _ encstruc (cindex, cb36);                                1c1f4
   type _ encstruc (cindex, cintrfrk);                           1c1f5
      #menu# !_ USE descrb (addkey (key, type));                 1c1f5a
         relent (cblock, type);                                  1c1f5a1
   type _ encstruc (cindex, cintrhst);                           1c1f6
      #menu# !_ USE descrb (addkey (key, type));                 1c1f6a
         relent (cblock, type);                                  1c1f6a1
   vchmnu _ encstruc (clist, $menu);                             1c1f7
      #menu# _;                                                  1c1f7a
         relent (cblock, key);                                   1c1f7a1
%create folders%                                                 1c1g
   nxtdef _ $dfldtbl;                                            1c1g1
   FOR i _ 1 UP UNTIL > cmfldn DO                                1c1g2
      BEGIN                                                      1c1g2a
      [nxtdef,dffldа] _ crtfld (nxtdef);                         1c1g2b
      nxtdef _ nxtdef + defr,SIZE;                               1c1g2c
      END;                                                       1c1g2d
%spawn self%                                                     1c1h
   spawn (0);                                                    1c1h1
```

```
%create self's process record%                                  1c1i
   ps _ crtrec (vpsfld, call : vfph);                           1c1i1
   INVOKE (prtnkp,, ps);                                        1c1i2
   ps.pstype _ cself;                                           1c1i3
   ps.psadjph _ cfph;                                           1c1i4
%create superior's process/port record%                         1c1j
   IF chntype _ vstupacs.LH THEN                                1c1j1
      BEGIN                                                     1c1j1a
      %create port record%                                      1c1j1b
         po _ crtrec (vpofld, call : poh);                      1c1j1b1
         INVOKE (prtnkp,, po);                                  1c1j1b2
         po.potype   _ chntype;                                 1c1j1b3
         po.poformat _ vstupacs.RH;                             1c1j1b4
         po.pofkh    _ -1;                                      1c1j1b5
      %allocate and create port%                                1c1j1c
         CASE chntype OF                                        1c1j1c1
            = cintrfrk;                                         1c1j1c1a
               BEGIN                                            1c1j1c1a1
               relent (cblock, falopo (po, 0));                 1c1j1c1a2
               fsndch (po, 0);                                  1c1j1c1a3
               END;                                             1c1j1c1a4
            = cintrhst, = cintrjob: sigerr (eefimp);            1c1j1c1b
            ENDCASE sigerr (ecutyp);                            1c1j1c1c
         po.pocreated _ TRUE;                                   1c1j1c2
      %create process record%                                   1c1j1d
```

```
                ps _ crtrec (vpsfld, call : vsph);                    1c1j1d1
                INVOKE (prtnkp,, ps);                                 1c1j1d2
                ps.pstype  _ csuperior;                               1c1j1d3
                ps.psadjph _ csph;                                    1c1j1d4
                ps.pspoh   _ poh;                                     1c1j1d5
             END                                                      1c1j1e
          ELSE                                                        1c1j2
             BEGIN                                                    1c1j2a
             vtrldr _ TRUE;                                           1c1j2b
             xinips (                                                 1c1j2c
                oiptstr (opntwrd ($vstupacs+1, CHAR),
                $intrahostaddr), 0, 0, 0);                            1c1j2c1
             END;                                                     1c1j2d
       %return%                                                       1c1k
          owait (0);                                                  1c1k1
          END.
                                                                      1c1k2
    (dpsrcvr) %recovery program%
    PROCEDURE (type, message REF, addr);                              1c2
       %output message at user's terminal%                            1c2a
          errmsg (101B, &message, addr);                              1c2a1
       %return%                                                       1c2b
          LOOP oendfk (0);                                            1c2b1
          END.
                                                                      1c2b2
    (err) %L10 run-time error%
    PROCEDURE (message REF);                                          1c3
```

```
        %advance meter%                                                 1c3a
           BUMP vmlerc;                                                 1c3a1
        %trace error%                                                   1c3b
           IF dtrace ,A ctrerrs THEN                                    1c3b1
              trcerr (eefl10, &message);                                1c3b1a
        %abort%                                                         1c3c
           ABORT (eefl10, &message);                                    1c3c1
           END,                                                         1c3c2

  %managers%                                                            1d
     (oprmgr) %operation manager%
     PROCEDURE (pcrh);                                                  1d1
        %signals%                                                       1d1a
           % NOTE (crelpr) %                                           1d1a1
        %declarations%                                                  1d1b
           LOCAL op, resumed=FALSE;                                     1d1b1
           LOCAL acs [cnoacs];                                          1d1b2
           LOCAL pr POINTER;                                            1d1b3
        %catchphrases%                                                  1d1c
           (punlckpr) CATCHPHRASE (: vdgmsg); CASE TRUE OF              1d1c1
              = abr ();                                                 1d1c1a
                 BEGIN                                                  1d1c1a1
                 %save error number for processor%                      1d1c1a2
                    acs [1] _ SIGNAL;                                   1d1c1a2a
                 %save diagnostic message for processor%                1d1c1a3
                    acs [2] _ 0;                                        1d1c1a3a
```

```
                IF vdgmsg AND pr,prrbaddr THEN                  1d1c1a3b
                  BEGIN                                         1d1c1a3b1
                  inirb (pr);                                   1d1c1a3b2
                  acs [2] _ wrpr (ccharstr, pr, vdgmsg := 0);   1d1c1a3b3
                  trmrb (pr);                                   1d1c1a3b4
                  END;                                          1d1c1a3b5
              %resume processor%                                1d1c1a4
                IF NOT resumed := TRUE THEN                     1d1c1a4a
                  rsmpr (pr, cfailure, $acs);                   1d1c1a4a1
                TERMINATE;                                      1d1c1a4b
              END;                                              1d1c1a5
            = rtn (); unlckrec (pr, ckeep);                     1d1c1b
            = (SIGNALTYPE = notetype AND SIGNAL = crelpr);      1d1c1c
              IF NOT resumed := TRUE THEN                       1d1c1c1
                rsmpr (pr, csuccess, $acs);                     1d1c1c1a
          ENDCASE;                                              1d1c1d
        %advance meter%                                         1d1d
          BUMP vmoprc;                                          1d1d1
        %establish manager context%                             1d1e
          vcch   _ 0;                                           1d1e1
          vcph   _ vfph;                                        1d1e2
          vcpcrh _ pcrh;                                        1d1e3
        %locate processor record%                               1d1f
          pr _ fndrec (vprfld, pcrh, cshare);                   1d1f1
          INVOKE (punlckpr, RETURN);                            1d1f2
```

```
        pr.prrunning _ FALSE;                                   1d1f3
        vcsph    _ pr.prsph;                                    1d1f4
        vcspldr _ pr.prspldr;                                   1d1f5
      %fetch contents of processor's ACs%                       1d1g
        ordacs (pr.prfkh, $acs);                                1d1g1
      %execute requested operation%                             1d1h
        IF NOT ((op _ acs.LH) IN [0, cmoprn]) THEN              1d1h1
          sigerr (eruopn);                                      1d1h1a
        [doprtbl [op]] (pr, $acs, 0);                           1d1h2
      %resume processor%                                        1d1i
        NOTE (crelpr);                                          1d1i1
      %return%                                                  1d1j
        RETURN;                                                 1d1j1
        END.                                                    1d1j2

(msgmgr) %message manager%
PROCEDURE (ph);                                                 1d2
    %declarations%                                              1d2a
      LOCAL routeph, srcph, dstph, dstsh, dstdead;              1d2a1
      LOCAL LIST msg [cmmsgl];                                  1d2a2
      LOCAL ps POINTER, sg POINTER, entry POINTER;              1d2a3
    %catchphrases%                                              1d2b
      (pnulst)  CATCHPHRASE; IF abrtn () THEN NULL-LISTS;       1d2b1
      (perrmsg) CATCHPHRASE (: vdgmsg); IF abr () THEN          1d2b2
        BEGIN                                                   1d2b2a
        #msg# _ 0, cmerror, SIGNAL, vdgmsg;                     1d2b2b
```

```
        sndps (srcph, $msg);                                    1d2b2c
        END;                                                    1d2b2d
    %assure local lists released%                               1d2c
      INVOKE (pnulst);                                          1d2c1
    %read message%                                              1d2d
      rcvps (srcph _ ph, $msg);                                 1d2d1
      INVOKE (perrmsg);                                         1d2d2
    %determine message's recipient%                             1d2e
      CASE (routeph _ ELEM #msg# [cmrout]) OF                   1d2e1
        = cfph: %from neighbor%                                 1d2e1a
          BEGIN                                                 1d2e1a1
          %establish manager context%                           1d2e1a2
            vctx _ 0;                                           1d2e1a2a
            oblkxfr ($vctx, $vctx+1, cmctxl-1);                 1d2e1a2b
            vcph _ srcph;                                       1d2e1a2c
          %process message%                                     1d2e1a3
            entry _ fndmsg (ELEM #msg# [cmnumb]);               1d2e1a3a
            #msg# _ MOVE #msg# [cmhdrc+1 TO msg,L];             1d2e1a3b
            l10cal (entry, $msg, 0);                            1d2e1a3c
          END;                                                  1d2e1a4
        < csgmnh: %via neighbor%                                1d2e1b
          BEGIN                                                 1d2e1b1
          %locate process record%                               1d2e1b2
            ps _ fndrec (vpsfld, routeph, cshare);              1d2e1b2a
            INVOKE (palwkp,, ps);                               1d2e1b2b
```

```
                IF ps.pstype # cintroduced                          1d2e1b2c
                OR ps.psadjph # srcph THEN sigerr (egmhsg);         1d2e1b2d
              %force consumption%                                   1d2e1b3
                srcph _ routeph;                                    1d2e1b3a
                REPEAT CASE (cfph);                                 1d2e1b3b
              END;                                                  1d2e1b4
            ENDCASE %relay%                                         1d2e1c
              BEGIN                                                 1d2e1c1
              %locate segment record%                               1d2e1c2
                sg _ fndrec (vsgfld, routeph, cshare);              1d2e1c2a
                INVOKE (palwkp,, sg);                               1d2e1c2b
              %verify logical channel alive%                        1d2e1c3
                dstph _ chndir (sg, srcph : dstsh, dstdead);        1d2e1c3a
                IF dstdead THEN sigerr (esfded);                    1d2e1c3b
              %forward message%                                     1d2e1c4
                #msg# [cmrout] _ dstsh;                             1d2e1c4a
                sndps (dstph, $msg);                                1d2e1c4b
              END;                                                  1d2e1c5
      %return%                                                      1d2e
        RETURN;                                                     1d2f1
        END.                                                        1d2f2
%+PDP10% %operations%                                               1e
  (ivdps) %invoke system call%
  PROCEDURE (pr POINTER, acs REF);                                  1e1
    %signals%                                                       1e1a
```

```
    % NOTE (crelpr) %                                                    1e1a1
  %declarations%                                                         1e1b
    LOCAL                                                                1e1b1
      evh, number, cch, syh, code, argc, resc,
      blocking=TRUE;                                                     1e1b1a
    LOCAL parms [cmsmlblk];                                              1e1b2
    LOCAL argl REF, resl REF;                                            1e1b3
    LOCAL entry POINTER, sy POINTER;                                     1e1b4
  %catchphrases%                                                         1e1c
    (punlcksy) CATCHPHRASE; IF abrtn () THEN                             1e1c1
      BEGIN                                                              1e1c1a
      %prepare for subsequent RRDPS%                                     1e1c1b
        IF NOT blocking THEN                                             1e1c1b1
          BEGIN                                                          1e1c1b1a
          %signal processor%                                             1e1c1b1b
            code,LH _ syh; code,RH _ number;                             1e1c1b1b1
            sigev (evh, code, TRUE);                                     1e1c1b1b2
          %save abort information%                                       1e1c1b1c
            IF abr () THEN                                               1e1c1b1c1
              BEGIN                                                      1e1c1b1c1a
              sy,syabort _ TRUE;                                         1e1c1b1c1b
              #resl# _ SIGNAL, vdgmsg;                                   1e1c1b1c1c
              END;                                                       1e1c1b1c1d
          END;                                                           1e1c1b1d
      %release system call record%                                       1e1c1c
```

```
            unlckrec (sy, IF (rtn () AND NOT blocking) OR
            SIGNAL = erorsb THEN ckeep ELSE cdelete);          1e1c1c1

         END;                                                  1e1c1d

      (ptrcabr)  CATCHPHRASE (: vdgmsg);                       1e1c2

         %trace system call abort%                             1e1c2a

            IF dtrace .A ctrsabr AND abr () THEN               1e1c2a1

               trcpr (csyc, number, entry, cabrlist, &resl);   1e1c2a1a

   %advance meter%                                             1e1d

      BUMP vmvjsc;                                             1e1d1

   %fetch operation parameters%                                1e1e

      rdpr (csmlblk, pr, acs.RH := 0, $parms);                 1e1e1

      evh _     parms.LH; number      _     parms.RH;          1e1e2

      cch _ parms [1].LH; pr.prrbaddr _ parms [1].RH;          1e1e3

   %lookup system call%                                        1e1f

      IF NOT (number IN [1, cmsycn]) THEN sigerr (erusyc);     1e1f1

      IF number # csipr AND NOT pr.prsignin THEN sigerr
      (erisin);                                                1e1f2

      entry _ sdsyctbl + 6*(number-1);                         1e1f3

   %verify controlling call handle%                            1e1g

      IF cch THEN INVOKE (palwkp,, (                           1e1g1

         fndcall (clocal, cch, cshare : vcch)));               1e1g1a

   %create call record%                                        1e1h

      sy _ crtrec (vsyfld, cprocessor : syh);                  1e1h1

      INVOKE (punlcksy);                                       1e1h2

      sy.synumb  _ number;                                     1e1h3

      sy.syentry _ entry;                                      1e1h4
```

```
        acs _ syh;                                                      1e1h5
    %allocate argument and result lists%                                1e1i
        sy.syargl _ &argl _ aloent (clist, argc _ entry.enargc);        1e1i1
        sy.syresl _ &resl _ aloent (clist, resc _ entry.enresc);        1e1i2
    %decode system call arguments%                                      1e1j
        decpr (pr, &acs, &argl, argc, entry+2);                         1e1j1
    %release processor%                                                 1e1k
        IF NOT (blocking _ NOT evh) THEN NOTE (crelpr);                 1e1k1
    %trace system call arguments%                                       1e1l
        IF dtrace .A ctrsarg THEN                                       1e1l1
            BEGIN                                                       1e1l1a
            trcpr (csyc, number, entry, carglist, &argl);               1e1l1b
            INVOKE (ptrcabr);                                           1e1l1c
            END;                                                        1e1l1d
    %execute system call%                                               1e1m
        l10cal (entry, &argl, &resl);                                   1e1m1
    %deliver system call results%                                       1e1n
        IF blocking THEN rrdps (pr, &acs, syh);                         1e1n1
    %return%                                                            1e1o
        RETURN;                                                         1e1o1
        END.                                                            1e1o2

(rrdps) %retrieve results of system call%
PROCEDURE (pr POINTER, acs REF, optsyh);                                1e2
    %declarations%                                                      1e2a
        LOCAL syh;                                                      1e2a1
```

```
    LOCAL resl REF;                                                 1e2a2
    LOCAL entry POINTER, sy POINTER;                                1e2a3
%fetch operation parameters%                                        1e2b
    IF optsyh THEN syh _ optsyh                                     1e2b1
    ELSE                                                            1e2b2
      BEGIN                                                         1e2b2a
      syh          _ acs,RH;                                        1e2b2b
      pr,prrbaddr _ acs [1];                                        1e2b2c
      END;                                                          1e2b2d
%locate system call record%                                         1e2c
    sy _ fndrec (vsyfld, syh, cexclusive);                          1e2c1
    IF optsyh THEN INVOKE (palwkp,, sy)                             1e2c2
    ELSE INVOKE (prtndl,, sy);                                      1e2c3
%examine outcome of system call%                                    1e2d
    entry _ sy,syentry;                                             1e2d1
    &resl _ sy,syresl;                                              1e2d2
    IF sy,syabort THEN                                              1e2d3
      BEGIN                                                         1e2d3a
      %trace system call abort%                                     1e2d3b
        IF dtrace ,A ctrsabr THEN                                   1e2d3b1
          trcpr (csyc, sy,synumb, entry, cabrlist, &resl);
                                                                    1e2d3b1a
      %abort%                                                       1e2d3c
        ABORT (ELEM #resl# [1], ELEM #resl# [2]);                   1e2d3c1
      END                                                           1e2d3d
    ELSE                                                            1e2d4
```

```
            BEGIN                                                   1e2d4a
            %trace system call results%                             1e2d4b
               IF dtrace ,A ctrsres THEN                            1e2d4b1
                  trcpr (csyc, sy,synumb, entry, creslist, &resl);
                                                                    1e2d4b1a
            %encode system call results%                            1e2d4c
               encpr (pr, &acs, &resl, entry,enresc, entry+5);      1e2d4c1
            END;                                                    1e2d4d
         %return%                                                   1e2e
            RETURN;                                                 1e2e1
            END.
                                                                    1e2e2

      (drdps) %discard results of system call%
      PROCEDURE (pr POINTER, acs REF);                              1e3
         %declarations%                                             1e3a
            LOCAL syh;                                              1e3a1
            LOCAL sy POINTER;                                       1e3a2
         %fetch operation parameters%                               1e3b
            syh _ acs.RH;                                           1e3b1
         %delete system call record%                                1e3c
            sy _ fndrec (vsyfld, syh, cdelete);                     1e3c1
            INVOKE (prtndl,, sy);                                   1e3c2
         %return%                                                   1e3d
            RETURN;                                                 1e3d1
            END.
                                                                    1e3d2

      (gtdps) %get user call arguments from DPS%
      PROCEDURE (pr POINTER, acs REF, optush, optrbaddr);           1e4
```

```
%declarations%                                                  1e4a
   LOCAL ush, number;                                           1e4a1
   LOCAL argl REF;                                              1e4a2
   LOCAL entry POINTER, us POINTER;                             1e4a3
%fetch operation parameters%                                    1e4b
   IF optush THEN                                               1e4b1
      BEGIN                                                     1e4b1a
      ush          _ optush;                                    1e4b1b
      pr,prrbaddr _ optrbaddr;                                  1e4b1c
      END                                                       1e4b1d
   ELSE                                                         1e4b2
      BEGIN                                                     1e4b2a
      ush          _ acs,RH;                                    1e4b2b
      pr,prrbaddr _ acs [1];                                    1e4b2c
      END;                                                      1e4b2d
%locate user call record%                                       1e4c
   us _ fndrec (vusfld, ush, cinternal);                        1e4c1
   INVOKE (palwkp,, us);                                        1e4c2
   IF us,uspcrh # vcpcrh THEN sigerr (egmhsg);                  1e4c3
%route errors to invoking manager%                              1e4d
   INVOKE (pnotmgr,, us);                                       1e4d1
%verify user call state%                                        1e4e
   IF us,usstate # cuncalled THEN sigerr (erigtd);              1e4e1
%trace user call arguments%                                     1e4f
   number _ us,usnumber;                                        1e4f1
```

```
        entry _ us,usentry;                                          1e4f2
        &argl _ us,usargl;                                           1e4f3
        IF dtrace ,A ctruarg THEN                                    1e4f4
           trcpr (cusc, number, entry, carglist, &argl);             1e4f4a
     %encode user call arguments%                                    1e4g
        encpr (pr, &acs, &argl, entry.enargc, entry+2);              1e4g1
     %advance user call state%                                       1e4h
        acs,LH _ number; acs,RH _ us,uscph;                          1e4h1
        us,usstate _ crunning;                                       1e4h2
     %return%                                                        1e4i
        RETURN;                                                      1e4i1
        END.                                                         1e4i2

(ptdps) %return user call results to DPS%
PROCEDURE (pr POINTER, acs REF, optush, optoutcome);                 1e5
     %declarations%                                                  1e5a
        LOCAL outcome, ush, state, dgmsg;                            1e5a1
        LOCAL parms [cmsmlblk];                                      1e5a2
        LOCAL resl REF;                                              1e5a3
        LOCAL entry POINTER, us POINTER;                             1e5a4
     %fetch operation parameters%                                    1e5b
        IF optush THEN                                               1e5b1
           BEGIN                                                     1e5b1a
           ush     _ optush;                                         1e5b1b
           outcome _ optoutcome;                                     1e5b1c
           END                                                       1e5b1d
```

```
      ELSE                                                          1e5b2
         BEGIN                                                      1e5b2a
         rdpr (csmlblk, pr, acs,RH, $parms);                        1e5b2b
         outcome _ parms,LH; ush _ parms,RH;                        1e5b2c
         END;                                                       1e5b2d
      %locate user call record%                                     1e5c
      us _ fndrec (vusfld, ush, cinternal);                         1e5c1
      INVOKE (palwkp,, us);                                         1e5c2
      IF us,uspcrh # vcpcrh THEN sigerr (egmhsg);                   1e5c3
      %route errors to invoking manager%                            1e5d
      INVOKE (pnotmgr,, us);                                        1e5d1
      %verify user call state%                                      1e5e
      IF (state _ us,usstate) # crunning                            1e5e1
      AND (state # cuncalled OR NOT outcome) THEN                   1e5e2
         sigerr (eriptd);                                           1e5e2a
      %decode user call results%                                    1e5f
      &res1 _ us,usres1;                                            1e5f1
      IF NOT outcome THEN                                           1e5f2
         BEGIN                                                      1e5f2a
         %decode results%                                           1e5f2b
            entry _ us,usentry;                                     1e5f2b1
            decpr (pr, &acs, &res1, entry,enresc, entry+4);         1e5f2b2
         %trace user call results%                                  1e5f2c
            IF dtrace ,A ctrures THEN                               1e5f2c1
```

```
                trcpr (cusc, us,usnumber, entry, creslist,
                &resl);                                          1e5f2c1a

            END                                                  1e5f2d

         ELSE                                                    1e5f3

            BEGIN                                                1e5f3a

            %decode diagnostic%                                  1e5f3b

               us,usdgmsg _ dgmsg _ rdpr (ccharstr, pr, acs [1],
               0);                                               1e5f3b1

            %trace user call abort%                              1e5f3c

               IF dtrace ,A ctruabr THEN                         1e5f3c1

                  BEGIN                                          1e5f3c1a

                  #resl# _ outcome, dgmsg;                       1e5f3c1b

                  trcpr (cusc, us,usnumber, entry, cabrlist,
                  &resl);                                        1e5f3c1c

                  END;                                           1e5f3c1d

            END;                                                 1e5f3d

      %notify invoking manager%                                  1e5g

         us,usstate _ ccompleted;                                1e5g1

         sigecb (us,usecb, IF outcome THEN outcome ELSE cok);    1e5g2

      %ready processor%                                          1e5h

         IF pr,prautordy THEN rdypr ();                          1e5h1

      %return%                                                   1e5i

         RETURN;                                                 1e5i1

         END,                                                    1e5i2

(pgdps) %return/get user call results/arguments%
PROCEDURE (pr POINTER, acs REF);                                 1e6
```

```
    %declarations%                                                  1e6a
      LOCAL outcome, rbaddr, ush;                                   1e6a1
      LOCAL parms [cmsmlblk];                                       1e6a2
    %fetch operation parameters%                                    1e6b
      rdpr (csmlblk, pr, acs,RH, $parms);                           1e6b1
      outcome _ parms,LH; rbaddr _ parms,RH;                        1e6b2
    %accept previous user call's results%                           1e6c
      IF ush _ pr,prush THEN ptdps (pr, &acs, ush, outcome);        1e6c1
    %wait for next user call%                                       1e6d
      pr,prush _ ush _ waiecb (pr,prsiecb);                         1e6d1
    %return next user call's results%                               1e6e
      gtdps (pr, &acs, ush, rbaddr);                                1e6e1
    %return%                                                        1e6f
      RETURN;                                                       1e6f1
      END,
                                                                    1e6f2

(erdps) %retrieve diagnostic message for DPS error%
PROCEDURE (pr POINTER, acs REF);                                    1e7
    %declarations%                                                  1e7a
      LOCAL number;                                                 1e7a1
    %fetch operation parameters%                                    1e7b
      number      _ acs,RH;                                         1e7b1
      pr,prrbaddr _ acs [1];                                        1e7b2
    %encode diagnostic message%                                     1e7c
      inirb (pr);                                                   1e7c1
      acs [1] _ wrpr (ccharstr, pr, fnderm (number));               1e7c2
```

```
        trmrb (pr);                                                 1e7c3

      %return%                                                      1e7d

        RETURN;                                                     1e7d1

        END.%+PDP10%                                                1e7d2

%system calls%                                                      1f

  %remote process manipulation%                                     1f1

    %processes%                                                     1f1a

      (crtps) %create remote process%
      PROCEDURE (psaddr REF, userinfo REF, stupinfo REF, scope,
      pknames REF, pkstupinfos REF, pkscope, pkhs REF %=> ph,
      pkhs REF%);                                                   1f1a1

        %declarations%                                              1f1a1a

          LOCAL ph, intpkinfo=0;                                    1f1a1a1

          LOCAL STRING intrahostaddr [cmiahal];                     1f1a1a2

          LOCAL LIST pkinfo [3];                                    1f1a1a3

          LOCAL intpkhs REF;                                        1f1a1a4

          LOCAL ps POINTER;                                         1f1a1a5

        %create process record%                                     1f1a1b

          ps _ crtrec (vpsfld, scope : ph);                         1f1a1b1

          INVOKE (prtnkp,, ps);                                     1f1a1b2

          ps.pstype  _ cinferior;                                   1f1a1b3

          ps.psadjph _ ph;                                          1f1a1b4

        %create process%                                            1f1a1c

          ps.pspoh _                                                1f1a1c1

            icrtps (&psaddr, &userinfo, &stupinfo, scope,
            $intrahostaddr);                                        1f1a1c1a
```

```
        %initialize process%                                        1f1a1d

           chgrec (ps, cshare);                                     1f1a1d1

           IF pkscope THEN                                          1f1a1d2

              BEGIN                                                 1f1a1d2a

              #pkinfo# != &pknames, &pkstupinfos, pkscope;          1f1a1d2b

              intpkinfo = $pkinfo;                                  1f1a1d2c

              END;                                                  1f1a1d2d

           syscal (ph, cxinips, $intrahostaddr, &userinfo,
           &stupinfo, vfpsdesc, intpkinfo, &pkhs :
           ps,pspsdesc, &intpkhs);                                  1f1a1d3

           IF pkscope THEN #pkhs# = MOVE #intpkhs#;                 1f1a1d4

        %return%                                                    1f1a1e

           RETURN (ph, IF pkhs.L THEN &pkhs ELSE 0);                1f1a1e1

           END,
                                                                    1f1a1e2

     (delps) %delete remote process%
     PROCEDURE (ph %=> cost%);                                      1f1a2

        %declarations%                                              1f1a2a

           LOCAL nxtph, port, ch, nxtcost, cost=0,
           outcome=TRUE;                                            1f1a2a1

           LOCAL ps POINTER;                                        1f1a2a2

        %delete one process%                                        1f1a2b

           IF ph THEN                                               1f1a2b1

              BEGIN                                                 1f1a2b1a

              %locate process record%                               1f1a2b1b

                 ps = fndrec (vpsfld, ph, cdelete);                 1f1a2b1b1

                 INVOKE (prtndl,, ps);                              1f1a2b1b2
```

```
            %verify request%                                    1f1a2b1c

               IF ps.pstype # cinferior THEN sigerr
               (esisup);                                        1f1a2b1c1

            %flush process from system%                         1f1a2b1d

               psflsh (ph);                                     1f1a2b1d1

            %delete process%                                    1f1a2b1e

               IF NOT ps.psdead THEN                            1f1a2b1e1

                  BEGIN                                         1f1a2b1e1a

                  syscal (ph, cxtrmps : cost);                  1f1a2b1e1b

                  idelps (ps,pspoh);                            1f1a2b1e1c

                  END;                                          1f1a2b1e1d

            END                                                 1f1a2b1f

         %delete all processes%                                  1f1a2c

            ELSE                                                1f1a2c1

               BEGIN                                            1f1a2c1a

               OPENPORT runfld (vpsfld, cdelete ; [port]);      1f1a2c1b

               WHILE ps _ PCALL [port] (cprevue : nxtph) DO     1f1a2c1c

                  IF ps.pstype = cinferior                      1f1a2c1c1

                  AND pCALL [port] (clock) THEN                 1f1a2c1c2

                     BEGIN                                      1f1a2c1c2a

                     nxtcost _ delps (nxtph ;pfail,, $outcome);
                                                                1f1a2c1c2b
                     IF outcome := TRUE THEN                    1f1a2c1c2c

                        cost _ cost + nxtcost;                  1f1a2c1c2c1

                     END;                                       1f1a2c1c2d

               END;                                             1f1a2c1d
```

```
    %return%                                                    1f1a2d
      RETURN (cost);                                            1f1a2d1
      END.
                                                                1f1a2d2

  %+PDP10% (itdps) %introduce remote processes%
  PROCEDURE (ph1, stupinfo1 REF, ph2, stupinfo2 REF, scope,
  flags %-> flags+ih, ph12, ph21,%);                            1f1a3
    %declarations%                                              1f1a3a
      LOCAL                                                     1f1a3a1
        logonly, sh, ch, adjph1, adjph2, ph12, ph21,
        newsh1, newsh2, outcome, pch;                           1f1a3a1a
      LOCAL ps1 POINTER, ps2 POINTER, sg POINTER;               1f1a3a2
    %catchphrases%                                              1f1a3b
      (pdelh1f1) CATCHPHRASE; IF abr () THEN                    1f1a3b1
        BEGIN                                                   1f1a3b1a
        IF ch THEN sysend (ch : newsh1);                        1f1a3b1b
        syscal (adjph1, cxdelchh, newsh1);                      1f1a3b1c
        END;                                                    1f1a3b1d
      (pdelh1f2) CATCHPHRASE; IF abr () THEN                    1f1a3b2
        syscal (adjph2, cxdelchh, newsh2);                      1f1a3b2a
    %isolate flags%                                             1f1a3c
      logonly _ flags ,A 4B11;                                  1f1a3c1
    %locate process record 1%                                   1f1a3d
      ps1 _ fndrec (vpsfld, ph1, cshare);                       1f1a3d1
      INVOKE (palwkp,, ps1);                                    1f1a3d2
    %locate process record 2%                                   1f1a3e
      ps2 _ fndrec (vpsfld, ph2, cshare);                       1f1a3e1
```

```
    INVOKE (palwkp,, ps2);                                          1f1a3e2

  %create new segment record%                                       1f1a3f

    sg _ crtrec (vsgfld, scope : sh);                               1f1a3f1

    INVOKE (prtnkp,, sg);                                           1f1a3f2

    sg.sgintro _ TRUE;                                              1f1a3f3

    sg.sgph1 _ adjph1 _ ps1.psadjph;                                1f1a3f4

    sg.sgph2 _ adjph2 _ ps2.psadjph;                                1f1a3f5

  %create channel halves in LH and RH processes%                    1f1a3g

    ch _                                                            1f1a3g1

      sysbgn (adjph1, cxcrtchh, ps2.pspsdesc,
      &stupinfo1, sh, ps1.psadjsh);                                 1f1a3g1a

    INVOKE (pdelh1f1);                                              1f1a3g2

    syscal (adjph2, cxcrtchh, ps1.pspsdesc, &stupinfo2,
    sh, ps2.psadjsh : newsh2, ph21);                                1f1a3g3

    sg.sgsh2 _ newsh2;                                              1f1a3g4

    INVOKE (pdelh1f2);                                              1f1a3g5

    sysend (ch := 0 : newsh1, ph12);                                1f1a3g6

    sg.sgsh1 _ newsh1;                                              1f1a3g7

  %create and associate physical with logical channel%              1f1a3h

    IF NOT logonly THEN                                             1f1a3h1

      BEGIN                                                         1f1a3h1a

      pch _                                                         1f1a3h1b

        crtch (ph1, ph12, ph2, ph21, scope ;pfail,,
        $outcome);                                                  1f1a3h1b1

      IF outcome THEN                                               1f1a3h1c

        BEGIN                                                       1f1a3h1c1
```

```
            sg,sgpch _ pch;                                    1f1a3h1c2
            sh,LH _ 4B11;                                      1f1a3h1c3
            END;                                               1f1a3h1c4
         END;                                                  1f1a3h1d
      %return%                                                 1f1a3i
         RETURN (sh, ph12, ph21);                              1f1a3i1
         END,%+PDP10%
                                                               1f1a3i2
%+PDP10% (sepps) %separate remote processes%
PROCEDURE (ih %-> cost1, cost2%);                              1f1a4
   %declarations%                                              1f1a4a
      LOCAL                                                    1f1a4a1
         pch, ch, dead, port, nxtih, nxtcost1, nxtcost2,
         cost1=0, cost2=0, outcome=TRUE;                       1f1a4a1a
      LOCAL sg POINTER;                                        1f1a4a2
   %separate one process%                                      1f1a4b
      IF ih THEN                                               1f1a4b1
         BEGIN                                                 1f1a4b1a
         %locate segment record%                               1f1a4b1b
            sg _ fndrec (vsgfld, ih, cdelete);                 1f1a4b1b1
            INVOKE (prtndl,, sg);                              1f1a4b1b2
            IF NOT sg,sgintro THEN sigerr (egmhsg);            1f1a4b1b3
         %delete physical channel if any%                      1f1a4b1c
            IF pch _ (sg,sgpch := 0) THEN delch (pch);         1f1a4b1c1
         %delete logical channel in adj processes%             1f1a4b1d
            IF NOT dead _ sg,sg1dead THEN ch _                 1f1a4b1d1
```

```
                sysbgn (sg.sgph1, cxdelchh, sg.sgsh1); 1f1a4b1d1a
             IF NOT sg.sg2dead THEN                    1f1a4b1d2
                syscal (sg.sgph2, cxdelchh, sg.sgsh2 :
                cost2);                                1f1a4b1d2a
             IF NOT dead THEN sysend (ch : cost1);     1f1a4b1d3
          END                                          1f1a4b1e
       %separate all processes%                        1f1a4c
       ELSE                                            1f1a4c1
          BEGIN                                        1f1a4c1a
          OPENPORT runfld (vsgfld, cdelete : [port]);  1f1a4c1b
          WHILE sg _ PCALL [port] (cprevue : nxtih) DO 1f1a4c1c
             IF sg.sgintro AND PCALL [port] (clock) THEN
                                                       1f1a4c1c1
                BEGIN                                  1f1a4c1c1a
                nxtcost1 _                             1f1a4c1c1b
                   sepps (nxtih : nxtcost2 ;pfail,,
                   soutcome);                          1f1a4c1c1b1
                IF outcome := TRUE THEN                1f1a4c1c1c
                   BEGIN                               1f1a4c1c1c1
                   cost1 _ cost1 + nxtcost1;           1f1a4c1c1c2
                   cost2 _ cost2 + nxtcost2;           1f1a4c1c1c3
                   END;                                1f1a4c1c1c4
                END;                                   1f1a4c1c1d
          END;                                         1f1a4c1d
       %return%                                        1f1a4d
       RETURN (cost1, cost2);                          1f1a4d1
```

```
        END.%+PDP10%                                                    1f1a4d2

    %+PDP10% (infps) %retrieve information about remote
    process%
    PROCEDURE (ph, type %-> info%);                                     1f1a5

       %declarations%                                                   1f1a5a

          LOCAL info;                                                   1f1a5a1

          LOCAL LIST psdesc [cmpdcl];                                   1f1a5a2

          LOCAL ps POINTER;                                             1f1a5a3

       %catchphrases%                                                   1f1a5b

          (pnulst) CATCHPHRASE; IF abrtn () THEN NULL-LISTS;            1f1a5b1

       %assure local lists released%                                    1f1a5c

          INVOKE (pnulst);                                              1f1a5c1

       %locate process record%                                          1f1a5d

          ps _ fndrec (vpsfld, ph, cshare);                             1f1a5d1

          INVOKE (palwkp,, ps);                                         1f1a5d2

       %retrieve information%                                           1f1a5e

          CASE type OF                                                  1f1a5e1

             = chostaddr:                                               1f1a5e1a

                BEGIN                                                   1f1a5e1a1

                decstruc (cpsdesc, ps.pspsdesc, $psdesc);               1f1a5e1a2

                info _ ELEM #[ELEM #psdesc# [3]]# [1];                  1f1a5e1a3

                END;                                                    1f1a5e1a4

             ENDCASE sigerr (eruinf);                                   1f1a5e1b

       %return%                                                         1f1a5f

          RETURN (info);                                                1f1a5f1
```

```
        END,%+PDP10%                                                    1f1a5f2

    %packages%                                                          1f1b

        (opnpk) %open remote packages%
        PROCEDURE (ph, pknames REF, stupinfos REF, scope, pkhs
        REF %-> pkhs REF%);                                             1f1b1

            %declarations%                                              1f1b1a

                LOCAL intpkhs REF;                                      1f1b1a1

            %open remote packages%                                      1f1b1b

                syscal (                                                1f1b1b1

                    ph, cxopnpk, &pknames, &stupinfos, scope, &pkhs
                    : &intpkhs);                                        1f1b1b1a

                INVOKE (palwrl,, &intpkhs);                             1f1b1b2

                #pkhs# _ MOVE #intpkhs#;                                1f1b1b3

            %return%                                                    1f1b1c

                RETURN (&pkhs);                                         1f1b1c1

                END,
                                                                        1f1b1c2

        (clspk) %close remote packages%
        PROCEDURE (ph, pkhs REF, costs REF %-> costs REF%);             1f1b2

            %declarations%                                              1f1b2a

                LOCAL intcosts REF;                                     1f1b2a1

            %close remote packages%                                     1f1b2b

                syscal (ph, cxclspk, &pkhs, &costs : &intcosts);        1f1b2b1

                INVOKE (palwrl,, &intcosts);                            1f1b2b2

                #costs# _ MOVE #intcosts#;                              1f1b2b3

            %return%                                                    1f1b2c

                RETURN (&costs);                                        1f1b2c1
```

```
        END,                                                    1f1b2c2

%procedures%                                                    1f1c

   (calpe) %call remote procedure%
   PROCEDURE (psel REF, priority, argl REF, arglmsk REF,
   reslmsk REF, resl REF, entry POINTER %-> outcome, resl
   REF, cost%);                                                 1f1c1

      %declarations%                                            1f1c1a

         LOCAL ch, outcome, cost;                               1f1c1a1

      %catchphrases%                                            1f1c1b

         (prelch) CATCHPHRASE; IF abr () THEN relch (ch);       1f1c1b1

      %allocate call handle%                                    1f1c1c

         ch _ aloch (&psel, priority, entry);                   1f1c1c1

         INVOKE (prelch);                                       1f1c1c2

      %call remote procedure%                                   1f1c1d

         relpe (ch, &argl, &arglmsk, &reslmsk, 0);              1f1c1d1

         LOOP CASE outcome _ acqpe (ch, &resl) OF               1f1c1d2

            = cvisit: sigerr (epwvis);                          1f1c1d2a

            = csiged: NULL;                                     1f1c1d2b

            ENDCASE EXIT LOOP;                                  1f1c1d2c

      %release call handle%                                     1f1c1e

         cost _ relch (ch);                                     1f1c1e1

      %return%                                                  1f1c1f

         RETURN (outcome, &resl, cost);                         1f1c1f1

         END,                                                   1f1c1f2

   (vispe) %visit remote callee/caller%
```

```
PROCEDURE (ch, argl REF, arglmsk REF, reslmsk REF, resl
REF %-> outcome, resl REF%);                                    1f1c2

   %release control to remote procedure%                        1f1c2a

      relpe (ch, &argl, &arglmsk, &reslmsk, 0);                 1f1c2a1

   %return%                                                     1f1c2b

      RETURN (acqpe (ch, &resl), &resl);                        1f1c2b1

      END,
                                                                1f1c2b2

(aloch) %allocate call handle for remote procedure call%
PROCEDURE (psel REF, priority, entry POINTER %-> ch%);          1f1c3

   %declarations%                                               1f1c3a

      LOCAL ch, pkh;                                            1f1c3a1

      LOCAL ca POINTER;                                         1f1c3a2

   %create call record%                                         1f1c3b

      ca _ crtrec (vcafld, call : ch);                          1f1c3b1

      INVOKE (prtnkp,, ca);                                     1f1c3b2

      ca.castate  _ cuncalled;                                  1f1c3b3

      ca.cacch    _ vcch;                                       1f1c3b4

      ca.caremote _ TRUE;                                       1f1c3b5

      ca.caph     _ pshand (ELEM #psel# [1]);                   1f1c3b6

      ca.capkh    _ pkh _ ELEM #psel# [2];                      1f1c3b7

      ca.capname  _                                             1f1c3b8

         encstruc (IF pkh THEN ccharstr ELSE cindex, ELEM
         #psel# [3]);                                           1f1c3b8a

      ca.caprio   _ priority;                                   1f1c3b9

      IF NOT pkh THEN ca.caentry _ entry;                       1f1c3b10

   %return%                                                     1f1c3c
```

```
        RETURN (ch);                                                1f1c3c1

        END.
                                                                    1f1c3c2

    (relch) %[abort remote callee and] release call handle%
    PROCEDURE (ch %-> cost%);                                       1f1c4

      %declarations%                                                1f1c4a

        LOCAL                                                       1f1c4a1

          port, nxtch, state, nxtcost, cost=0,
          outcome=TRUE;                                             1f1c4a1a

        LOCAL ca POINTER;                                           1f1c4a2

      %release one procedure%                                       1f1c4b

        IF ch THEN                                                  1f1c4b1

          BEGIN                                                     1f1c4b1a

          %locate call record%                                      1f1c4b1b

            ca _ fndcall (cremote, ch, cinternal);                  1f1c4b1b1

            INVOKE (prtndl,, ca);                                   1f1c4b1b2

          %abort procedure%                                         1f1c4b1c

            state _ ca,castate;                                     1f1c4b1c1

            IF state # cuncalled AND state # ccompleted
            THEN                                                    1f1c4b1c2

              BEGIN                                                 1f1c4b1c2a

              %abort procedure%                                     1f1c4b1c2b

                syscal (ca,caph, cxabrpe, ch);                      1f1c4b1c2b1

              %wait for procedure to return%                        1f1c4b1c2c

                WHILE (outcome _ acqpe (ch, 0)) =
                cvisit OR outcome _ csiged DO NULL;
                                                                    1f1c4b1c2c1

              END;                                                  1f1c4b1c2d
```

```
            cost _ ca.cacost;                                   1f1c4b1c3
          END                                                   1f1c4b1d
      %release all procedures%                                  1f1c4c
        ELSE                                                    1f1c4c1
          BEGIN                                                 1f1c4c1a
          OPENPORT runfld (vcafld, cinternal : [port]);         1f1c4c1b
          WHILE ca _ PCALL [port] (cprevue : nxtch) DO          1f1c4c1c
            IF ca.caremote AND PCALL [port] (clock) THEN
                                                                1f1c4c1c1
              BEGIN                                             1f1c4c1c1a
              nxtcost _ relch (nxtch ;pfail,, $outcome);
                                                                1f1c4c1c1b
              IF outcome := TRUE THEN cost _ cost +
              nxtcost;                                          1f1c4c1c1c
              END;                                              1f1c4c1c1d
          END;                                                  1f1c4c1d
      %return%                                                  1f1c4d
        RETURN (cost);                                          1f1c4d1
        END.
                                                                1f1c4d2

(acqpe) %acquire control from remote callee/caller%
PROCEDURE (ch, resl REF %-> outcome, resl REF%);                1f1c5
      %declarations%                                            1f1c5a
        LOCAL outcome;                                          1f1c5a1
        LOCAL intresl REF;                                      1f1c5a2
        LOCAL ca POINTER, entry POINTER;                        1f1c5a3
      %locate call record%                                      1f1c5b
        ca _ fndrec (vcafld, ch, cinternal);                    1f1c5b1
```

```
      INVOKE (palwkp,, ca);                                          1f1c5b2
   %verify call state%                                               1f1c5c
      CASE ca,castate OF                                             1f1c5c1
         = crunrem, = creturned:                                     1f1c5c1a
            outcome _ waiecb (ca,caqecb);                            1f1c5c1a1
         ENDCASE sigerr (epiacq);                                    1f1c5c1b
   %claim procedure results%                                         1f1c5d
      &intresl _ (ca,caprml := 0);                                   1f1c5d1
      INVOKE (palwrl,, &intresl);                                    1f1c5d2
      IF &resl AND &intresl AND outcome # caborted THEN              1f1c5d3
         #resl# _ MOVE #intresl#;                                    1f1c5d3a
   %examine outcome%                                                 1f1c5e
      CASE outcome OF                                                1f1c5e1
         = csiged: sigecb (ca,carecb, cok);                          1f1c5e1a
         = cvisit: ca,castate _ crunloc;                             1f1c5e1b
         = csuccess, = cfailure:                                     1f1c5e1c
            BEGIN                                                    1f1c5e1c1
            ca,castate _ ccompleted;                                 1f1c5e1c2
            IF entry _ ca,caentry THEN                               1f1c5e1c3
               decprms (&resl, entry,enresc, entry+4);
                                                                     1f1c5e1c3a
            END;                                                     1f1c5e1c4
         = caborted:                                                 1f1c5e1d
            BEGIN                                                    1f1c5e1d1
            ca,castate _ ccompleted;                                 1f1c5e1d2
            ABORT (                                                  1f1c5e1d3
```

```
                decstruc (cindex,   ELEM #intresl# [1],
                0),                                          1f1c5e1d3a

                decstruc (ccharstr, ELEM #intresl# [2],
                $verrmsg));                                  1f1c5e1d3b

            END;                                             1f1c5e1d4

        ENDCASE sigerr (epuotc);                             1f1c5e1e

    %return%                                                 1f1c5f

        RETURN (outcome, &resl);                             1f1c5f1

        END.
                                                             1f1c5f2

(relpe) %release control to remote callee/caller%
PROCEDURE (ch, argl REF, arglmsk REF, reslmsk REF,
acqevh);                                                     1f1c6

    %declarations%                                           1f1c6a

        LOCAL LIST msg [cmmsgl];                             1f1c6a1

        LOCAL entry POINTER, ca POINTER;                     1f1c6a2

    %catchphrases%                                           1f1c6b

        (pnulst) CATCHPHRASE; IF abrtn () THEN NULL-LISTS;   1f1c6b1

    %assure local lists released%                            1f1c6c

        INVOKE (pnulst);                                     1f1c6c1

    %locate call record%                                     1f1c6d

        ca _ fndrec (vcafld, ch, cinternal);                 1f1c6d1

        INVOKE (palwkp,, ca);                                1f1c6d2

        ca.caqevh _ acqevh;                                  1f1c6d3

    %construct message%                                      1f1c6e

        #msg# _ 0;                                           1f1c6e1

        CASE ca.castate OF                                   1f1c6e2
```

```
            = cuncalled;                                    1f1c6e2a
               BEGIN                                        1f1c6e2a1
               IF entry _ ca.caentry THEN                   1f1c6e2a2
                  encprms (&argl, entry.enargc, entry+2, 0);
                                                            1f1c6e2a2a
               #msg# !_                                     1f1c6e2a3
                  cmcalpe, ch, ca.capkh, ca.capname, &argl,
                  &arglmsk, &reslmsk, ca.caprio;            1f1c6e2a3a
               END;                                         1f1c6e2a4
            = crunloc;                                      1f1c6e2b
               BEGIN                                        1f1c6e2b1
               mskarlst (creslist, &argl, ca.camskl);       1f1c6e2b2
               relent (clist, ca.camskl := 0);              1f1c6e2b3
               IF ca.caremote THEN                          1f1c6e2b4
                  #msg# !_                                  1f1c6e2b4a
                     cmrecpe, ch, &argl, &arglmsk, &reslmsk
                                                            1f1c6e2b4a1
                  ELSE                                      1f1c6e2b5
                     #msg# !_                               1f1c6e2b5a
                        cmrtnpe, ca.cacch, &argl, &arglmsk,
                        &reslmsk, cvisit, 0;                1f1c6e2b5a1
               END;                                         1f1c6e2b6
            ENDCASE sigerr (epirel);                        1f1c6e2c
         %send CALPE/RECPE/RTNPE message%                   1f1c6f
            ca.caqevh _ acqevh;                             1f1c6f1
            sndps (ca.caph, $msg);                          1f1c6f2
            ca.castate _ crunrem;                           1f1c6f3
         %return%                                           1f1c6g
```

```
        RETURN;                                                 1f1c6g1

        END,
                                                                1f1c6g2

(sigpe) %signal remote callee/caller%
PROCEDURE (ch, argl REF, arglmsk REF);                          1f1c7

    %signals%                                                   1f1c7a

        % NOTE (crelpr) %                                       1f1c7a1

    %declarations%                                              1f1c7b

        LOCAL ca POINTER;                                       1f1c7b1

    %locate  call record%                                       1f1c7c

        ca _ fndrec (vcafld, ch, cinternal);                    1f1c7c1

        INVOKE (palwkp,, ca);                                   1f1c7c2

    %verify call state%                                         1f1c7d

        IF ca.castate # crunloc THEN sigerr (episig);           1f1c7d1

    %resume processor%                                          1f1c7e

        NOTE (crelpr);                                          1f1c7e1

    %signal%                                                    1f1c7f

        syscal (ca.caph, cxsigpe, IF ca.caremote THEN ch
        ELSE ca.cacch, &argl, &arglmsk);                        1f1c7f1

    %return%                                                    1f1c7g

        RETURN;                                                 1f1c7g1

        END,
                                                                1f1c7g2

(intpe) %interrupt remote callee%
PROCEDURE (ch);                                                 1f1c8

    %declarations%                                              1f1c8a

        LOCAL state, port, nxtch;                               1f1c8a1
```

```
    LOCAL ca POINTER;                                          1f1c8a2
%interrupt one procedure%                                      1f1c8b
    IF ch THEN                                                 1f1c8b1
      BEGIN                                                    1f1c8b1a
      %locate call record%                                     1f1c8b1b
        ca _ fndcall (cremote, ch, cinternal);                 1f1c8b1b1
        INVOKE (palwkp,, ca);                                  1f1c8b1b2
      %verify call state%                                      1f1c8b1c
        IF (state _ ca.castate) = cinterrupted THEN
                                                               1f1c8b1c1
          sigerr (epiint);                                     1f1c8b1c1a
      %interrupt procedure%                                    1f1c8b1d
        IF state # cuncalled AND state # ccompleted
        THEN                                                   1f1c8b1d1
          syscal (ca.caph, cxintpe, ch ;pignuch);
                                                               1f1c8b1d1a
        ca.caprvstate _ (ca.castate := cinterrupted);
                                                               1f1c8b1d2
      END                                                      1f1c8b1e
%interrupt all procedures%                                     1f1c8c
    ELSE                                                       1f1c8c1
      BEGIN                                                    1f1c8c1a
      OPENPORT runfld (vcafld, cinternal : [port]);            1f1c8c1b
      WHILE ca _ PCALL [port] (cprevue : nxtch) DO             1f1c8c1c
        IF ca.caremote                                         1f1c8c1c1
        AND ca.castate # cinterrupted                          1f1c8c1c2
        AND PCALL [port] (clock) THEN                          1f1c8c1c3
          intpe (nxtch ;pignore);                              1f1c8c1c3a
```

```
            END;                                                    1f1c8c1d

        %return%                                                    1f1c8d

            RETURN;                                                 1f1c8d1

            END.                                                    1f1c8d2

    (rsmpe) %resume remote callee%
    PROCEDURE (ch);                                                 1f1c9

        %declarations%                                              1f1c9a

            LOCAL state, port, nxtch;                               1f1c9a1

            LOCAL ca POINTER;                                       1f1c9a2

        %resume one procedure%                                      1f1c9b

            IF ch THEN                                              1f1c9b1

                BEGIN                                               1f1c9b1a

                %locate call record%                                1f1c9b1b

                    ca _ fndcall (cremote, ch, cinternal);          1f1c9b1b1

                    INVOKE (palwKP,, ca);                           1f1c9b1b2

                %verify call state%                                 1f1c9b1c

                    IF ca.castate # cinterrupted THEN               1f1c9b1c1

                        sigerr (epirsm);                            1f1c9b1c1a

                %resume procedure%                                  1f1c9b1d

                    ca.castate _ state _ caprvstate;                1f1c9b1d1

                    IF  state # cuncalled                           1f1c9b1d2

                    AND state # ccompleted THEN                     1f1c9b1d3

                        syscal (ca,caph, cxrsmpe, ch ;pignuch);
                                                                    1f1c9b1d3a

                END                                                 1f1c9b1e

        %resume all procedures%                                     1f1c9c
```

```
        ELSE                                                    1f1c9c1
          BEGIN                                                 1f1c9c1a
          OPENPORT runfld (vcafld, cinternal : [port]);         1f1c9c1b
          WHILE ca _ PCALL [port] (cprevue : nxtch) DO          1f1c9c1c
            IF ca.caremote                                      1f1c9c1c1
            AND ca.castate = cinterrupted                       1f1c9c1c2
            AND PCALL [port] (clock) THEN                       1f1c9c1c3
              rsmpe (nxtch ;pignore);                           1f1c9c1c3a
          END;                                                  1f1c9c1d
      %return%                                                  1f1c9d
        RETURN;                                                 1f1c9d1
        END,
                                                                1f1c9d2

    (ntepe) %make event known to remote caller%
    PROCEDURE (event, desc REF);                                1f1c10
      %signals%                                                 1f1c10a
        % NOTE (crelpr) %                                       1f1c10a1
      %declarations%                                            1f1c10b
        LOCAL cch;                                              1f1c10b1
        LOCAL ca POINTER;                                       1f1c10b2
      %locate call record%                                      1f1c10c
        ca _ fndcall (clocal, vcch, cinternal : cch);           1f1c10c1
        INVOKE (palwkp,, ca);                                   1f1c10c2
      %verify call state%                                       1f1c10d
        IF ca.castate # crunloc THEN sigerr (epinte);           1f1c10d1
      %resume processor%                                        1f1c10e
```

```
            NOTE (crelpr);                                          1f1c10e1
         %send note%                                                1f1c10f
            syscal (ca,caph, cxntepe, cch, event, &desc);           1f1c10f1
         %return%                                                   1f1c10g
            RETURN;                                                 1f1c10g1
            END,                                                    1f1c10g2

      (hlppe) %solicit help from remote caller%
      PROCEDURE (pblm, desc REF %-> solution REF%);                 1f1c11
         %declarations%                                             1f1c11a
            LOCAL cch, solution;                                    1f1c11a1
            LOCAL ca POINTER;                                       1f1c11a2
         %locate call record%                                       1f1c11b
            ca _ fndcall (clocal, vcch, cinternal : cch);           1f1c11b1
            INVOKE (palwkp,, ca);                                   1f1c11b2
         %verify call state%                                        1f1c11c
            IF ca.castate # crunloc THEN sigerr (epihlp);           1f1c11c1
         %solicit help%                                             1f1c11d
            syscal (                                                1f1c11d1
               ca.caph, cxhlppe, cch, pblm, &desc : solution);      1f1c11d1a
         %return%                                                   1f1c11e
            RETURN (solution);                                      1f1c11e1
            END,                                                    1f1c11e2

   %data stores%                                                    1f1d
      %+PDP10% (crtdt) %create remote data store%
      PROCEDURE (dsel REF, value REF, scope);                       1f1d1
```

```
    %create remote data store%                                          1f1d1a
       syscal (                                                         1f1d1a1
          ELEM #dsel# [1], cxcrtdt, &dsel, &value, scope);              1f1d1a1a
    %return%                                                            1f1d1b
       RETURN;                                                          1f1d1b1
       END,%+PDP10%                                                     1f1d1b2

%+PDP10% (deldt) %delete remote data store%
PROCEDURE (dsel REF);                                                   1f1d2
    %delete remote data store%                                          1f1d2a
       syscal (ELEM #dsel# [1], cxdeldt, &dsel);                        1f1d2a1
    %return%                                                            1f1d2b
       RETURN;                                                          1f1d2b1
       END,%+PDP10%                                                     1f1d2b2

%+PDP10% (rddt)  %read remote data store%
PROCEDURE (dsel REF %-> value REF%);                                    1f1d3
    %declarations%                                                      1f1d3a
       LOCAL value;                                                     1f1d3a1
    %read remote data store%                                            1f1d3b
       syscal (ELEM #dsel# [1], cxrddt, &dsel : value);                 1f1d3b1
    %return%                                                            1f1d3c
       RETURN (value);                                                  1f1d3c1
       END,%+PDP10%                                                     1f1d3c2

%+PDP10% (wrdt)  %write remote data store%
PROCEDURE (dsel REF, value REF);                                        1f1d4
    %write remote data store%                                           1f1d4a
```

```
            syscal (ELEM #dsel# [1], cxwrdt, &dsel, &value);    1f1d4a1

         %return%                                               1f1d4b

            RETURN;                                             1f1d4b1

            END,%+PDP10%
                                                                1f1d4b2

      %+PDP10% (lckdt) %lock remote data store%
      PROCEDURE (dsel REF, type, scope, flags %-> dtlh%);       1f1d5

         %declarations%                                         1f1d5a

            LOCAL wait, dtlh;                                   1f1d5a1

         %isolate flags%                                        1f1d5b

            wait _ NOT (flags .A 4B11);                         1f1d5b1

         %lock remote data store%                               1f1d5c

            syscal (                                            1f1d5c1

               ELEM #dsel# [1], cxlckdt, &dsel, type, scope,
               wait : dtlh);                                    1f1d5c1a

         %return%                                               1f1d5d

            RETURN (dtlh);                                      1f1d5d1

            END,%+PDP10%
                                                                1f1d5d2

      %+PDP10% (ulkdt) %unlock remote data store%
      PROCEDURE (dsel REF, dtlh);                               1f1d6

         %unlock remote data store%                             1f1d6a

            syscal (ELEM #dsel# [1], cxulkdt, &dsel, dtlh);     1f1d6a1

         %return%                                               1f1d6b

            RETURN;                                             1f1d6b1

            END,%+PDP10%
                                                                1f1d6b2

   %channels%                                                   1f1e
```

```
%+PDP10% (crtch) %create channel between remote
processes%
PROCEDURE (ph1, ph12, ph2, ph21, scope %-> pch, poh1,
poh2%);                                                    1f1e1

   %declarations%                                          1f1e1a

      LOCAL pch, poh1, poh2, port1, port2, i, ch=0;        1f1e1a1

      LOCAL LIST                                           1f1e1a2

         storage [2], menu [cmcmnul], psdesc1 [cmpdcl],
         psdesc2 [cmpdcl];                                 1f1e1a2a

      LOCAL ps1 POINTER, ps2 POINTER, cn POINTER;          1f1e1a3

   %catchphrases%                                          1f1e1b

      (pnulst)    CATCHPHRASE; IF abrtn () THEN
      NULL-LISTS;                                          1f1e1b1

      (prelport2) CATCHPHRASE; IF abr () THEN              1f1e1b2

         BEGIN                                             1f1e1b2a

         IF ch THEN sysend (ch);                           1f1e1b2b

         syscal (ph2, cxrelpo, poh2);                      1f1e1b2c

         END;                                              1f1e1b2d

      (prelport1) CATCHPHRASE; IF abr () THEN              1f1e1b3

         syscal (ph1, cxrelpo, poh1);                      1f1e1b3a

   %assure local lists released%                           1f1e1c

      INVOKE (pnulst);                                     1f1e1c1

   %locate process record 1%                               1f1e1d

      ps1 _ fndrec (vpsfld, ph1, cshare);                  1f1e1d1

      INVOKE (palwkp,, ps1);                               1f1e1d2

      decstruc (cpsdesc, ps1,pspsdesc, $psdesc1);          1f1e1d3

   %locate process record 2%                               1f1e1e
```

```
      ps2 _ fndrec (vpsfld, ph2, cshare);                          1f1e1e1
      INVOKE (palwkp,, ps2);                                       1f1e1e2
      decstruc (cpsdesc, ps2,pspsdesc, $psdesc2);                  1f1e1e3
   %create channel record%                                         1f1e1f
      cn _ crtrec (vcnfld, scope : pch);                           1f1e1f1
      INVOKE (prtnkp,, cn);                                        1f1e1f2
      cn,cnph1 _ ph1;                                              1f1e1f3
      cn,cnph2 _ ph2;                                              1f1e1f4
   %negotiate channel type with ps 2, allocate port%               1f1e1g
      syscal (ph1, cxrdmnu, $menu);                                1f1e1g1
      syscal                                                       1f1e1g2
         (ph2, cxalopo, $menu, ELEM #psdesc1# [3], FALSE
         : poh2, i, port2);                                        1f1e1g2a
      #storage# !_ USE descrb (port2);                             1f1e1g3
      INVOKE (prelport2);                                          1f1e1g4
   %allocate a corresponding port in process 1%                    1f1e1h
      #menu# _ MOVE #menu# [i];                                    1f1e1h1
      syscal                                                       1f1e1h2
         (ph1, cxalopo, $menu, ELEM #psdesc2# [3], TRUE :
         poh1, i, port1);                                          1f1e1h2a
      #storage# !_ USE descrb (port1);                             1f1e1h3
      INVOKE (prelport1);                                          1f1e1h4
   %burrow a channel from both ends%                               1f1e1i
      ch _ sysbgn (ph2, cxcrtce, poh2, port1, ph21);               1f1e1i1
      syscal (ph1, cxcrtce, poh1, port2, ph12);                    1f1e1i2
      sysend (ch := 0);                                            1f1e1i3
```

```
        %return%                                                    1f1e1j

            RETURN (pch, (cn.cnpoh1 _ poh1), (cn.cnpoh2 _
            poh2));                                                 1f1e1j1

            END.%+PDP10%
                                                                    1f1e1j2

        %+PDP10% (delch) %delete channel between remote
        processes%
        PROCEDURE (pch);                                            1f1e2

            %declarations%                                          1f1e2a

            LOCAL ch, dead, port, nxtpch;                           1f1e2a1

            LOCAL cn POINTER;                                       1f1e2a2

            %delete one channel%                                    1f1e2b

            IF pch THEN                                             1f1e2b1

                BEGIN                                               1f1e2b1a

                %locate channel record%                             1f1e2b1b

                    cn _ fndrec (vcnfld, pch, cdelete);             1f1e2b1b1

                    INVOKE (prtndl,, cn);                           1f1e2b1b2

                %flush channel from system%                         1f1e2b1c

                    cnflsh (pch);                                   1f1e2b1c1

                %dissolve channel from both ends%                   1f1e2b1d

                    IF NOT dead _ cn.cn1dead THEN                   1f1e2b1d1

                        ch _ sysbgn (cn.cnph1, cxrelpo,
                        cn.cnpoh1);                                 1f1e2b1d1a

                    IF NOT cn.cn2dead THEN                          1f1e2b1d2

                        syscal (cn.cnph2, cxrelpo, cn.cnpoh2);      1f1e2b1d2a

                    IF NOT dead THEN sysend (ch);                   1f1e2b1d3

                END                                                 1f1e2b1e
```

```
            %delete all channels%                                   1f1e2c
               ELSE                                                 1f1e2c1
                  BEGIN                                             1f1e2c1a
                  OPENPORT runfld (vcnfld, cdelete : [port]);       1f1e2c1b
                  WHILE cn _ PCALL [port] (clock : nxtpch) DO       1f1e2c1c
                     delch (nxtpch ;pignore);                       1f1e2c1c1
                  END;                                              1f1e2c1d
            %return%                                                1f1e2d
               RETURN;                                              1f1e2d1
               END.%+PDP10%
                                                                    1f1e2d2
%local process manipulation%                                        1f2
   %subprocesses%                                                   1f2a
      (crtsp) %create local subprocess%
      PROCEDURE (spaddr REF, stupinfo REF, scope, priority,
      psldr %-> sph%);                                              1f2a1
         %declarations%                                             1f2a1a
            LOCAL sph;                                              1f2a1a1
            LOCAL su POINTER;                                       1f2a1a2
         %create subprocess record%                                 1f2a1b
            su _ crtrec (vsufld, scope : sph);                      1f2a1b1
            INVOKE (prtnkp,, su);                                   1f2a1b2
            su.susuaddr _ savent (ccharstr, &spaddr);               1f2a1b3
            su.supsldr  _ psldr;                                    1f2a1b4
            su.sumnnpr  _ cmnnpr;                                   1f2a1b5
            su.sumxnpr  _ cmxnpr;                                   1f2a1b6
```

```
        %create subprocess leader%                                      1f2a1c
           chgrec (su, cshare);                                         1f2a1c1
           su.supcrh _                                                  1f2a1c2
              crtpr (sph, &stupinfo, scope, priority, TRUE);            1f2a1c2a
           su.susignin _ TRUE;                                          1f2a1c3
        %return%                                                        1f2a1d
           RETURN (sph);                                                1f2a1d1
           END.
                                                                        1f2a1d2

     (delsp) %delete local subprocess%
     PROCEDURE (sph, okifldr %-> cost%);                                1f2a2
        %declarations%                                                  1f2a2a
           LOCAL                                                        1f2a2a1
              nxtsph, pcrh, ldrpcrh, port, nxtcost, cost=0,
              outcome=TRUE;                                             1f2a2a1a
           LOCAL su POINTER, pr POINTER;                                1f2a2a2
        %delete one subprocess%                                         1f2a2b
           IF sph THEN                                                  1f2a2b1
              BEGIN                                                     1f2a2b1a
              %locate subprocess record%                                1f2a2b1b
                 su _ fndrec (vsufld, sph, cdelete);                    1f2a2b1b1
                 INVOKE (prtndl,, su);                                  1f2a2b1b2
                 IF (su.supsldr AND NOT okifldr)                        1f2a2b1b3
                 OR sph = vcsph THEN sigerr (egmhsu);                   1f2a2b1b4
              %flush subprocess from system%                            1f2a2b1c
                 suflsh (sph);                                          1f2a2b1c1
```

```
            %delete processors%                                    1f2a2b1d
               ldrpcrh _ su,supcrh;                                1f2a2b1d1
               OPENPORT runfld (vprfld, cdelete : [port]);
                                                                   1f2a2b1d2
               WHILE pr _ PCALL [port] (cprevue : pcrh) DO
                                                                   1f2a2b1d3
                  IF pr,prsph = sph                               1f2a2b1d3a
                  AND pcrh # ldrpcrh                              1f2a2b1d3b
                  AND PCALL [port] (clock) THEN                   1f2a2b1d3c
                     delpr (pcrh, FALSE);                         1f2a2b1d3c1
               %delete subprocess leader%                          1f2a2b1e
               delpr (ldrpcrh, TRUE);                              1f2a2b1e1
               cost _ su,sucost;                                   1f2a2b1e2
               %delete encapsulator%                               1f2a2b1f
               IF pcrh _ su,suepcrh THEN sepfk (pcrh, TRUE);
                                                                   1f2a2b1f1
            END                                                    1f2a2b1g
         %delete all subprocesses%                                   1f2a2c
         ELSE                                                      1f2a2c1
            BEGIN                                                  1f2a2c1a
            %delete subprocesses%                                  1f2a2c1b
               OPENPORT runfld (vsufld, cdelete : [port]);
                                                                   1f2a2c1b1
               WHILE su _ PCALL [port] (cprevue : nxtsph) DO
                                                                   1f2a2c1b2
                  IF NOT su,supsldr                               1f2a2c1b2a
                  AND nxtsph # vcsph                              1f2a2c1b2b
                  AND PCALL [port] (clock) THEN                   1f2a2c1b2c
                     BEGIN                                        1f2a2c1b2c1
                     nxtcost _                                    1f2a2c1b2c2
```

```
                        delsp (nxtsph, FALSE ;pfail,,
                        $outcome);                          1f2a2c1b2c2a

                      IF outcome := TRUE THEN cost _ cost +
                      nxtcost;                              1f2a2c1b2c3

                      END;                                  1f2a2c1b2c4

              %delete process leader%                         1f2a2c1c

                IF okifldr THEN                               1f2a2c1c1

                   cost _ cost + delsp (vpsldr, TRUE);        1f2a2c1c1a

              END;                                            1f2a2c1d

          %return%                                            1f2a2d

             RETURN (cost);                                   1f2a2d1

             END,
                                                              1f2a2d2

    %processors%                                              1f2b

       %+PDP10% (crtpr) %create local processor%
       PROCEDURE (sph, stuPinfo REF, scope, priority, spldr %=>
       pcrh%);                                                1f2b1

          %declarations%                                      1f2b1a

             LOCAL pcrh, fkh, pg;                             1f2b1a1

             LOCAL su POINTER, pr POINTER;                    1f2b1a2

          %catchphrases%                                      1f2b1b

             (pdelfk) CATCHPHRASE; IF abr () THEN odelfk (fkh); 1f2b1b1

             (pendfk) CATCHPHRASE; IF abr () THEN oendfk (fkh); 1f2b1b2

          %locate subprocess record%                          1f2b1c

             su _ fndrec (vsufld, sph, cshare);               1f2b1c1

             INVOKE (palwkp,, su);                            1f2b1c2

          %create processor record%                           1f2b1d
```

```
      pr _ crtrec (vprfld, scope : pcrh);                    1f2b1d1
      INVOKE (prtnkp,, pr);                                  1f2b1d2
      pr.prstup  _ savent (cblock, &stupinfo);               1f2b1d3
      pr.prspldr _ spldr;                                    1f2b1d4
      pr.prsph   _ sph;                                      1f2b1d5
      pr.prprio  _ priority;                                 1f2b1d6
   %create processor fork%                                   1f2b1e
      pr.prfkh _ fkh _ ocrtfk (su.susuaddr);                 1f2b1e1
      INVOKE (pdelfk);                                       1f2b1e2
   %intercept DPS operations%                                1f2b1f
      otrpfk (fkh);                                          1f2b1f1
   %interconnect addr space with subprocess leader's%        1f2b1g
      IF spldr THEN su.sufkh _ fkh                           1f2b1g1
      ELSE                                                   1f2b1g2
         ocnnfk (su.sufkh, pg, fkh, pg _ su.supg,
         su.supgcnt);                                        1f2b1g2a
   %start processor%                                         1f2b1h
      obgnfk (fkh);                                          1f2b1h1
      INVOKE (pendfk);                                       1f2b1h2
   %wait for processor to sign in%                           1f2b1i
      chgrec (pr, cshare);                                   1f2b1i1
      waicok (pr.prsiecb);                                   1f2b1i2
   %bump processor count%                                    1f2b1j
      BUMP su.sucrnpr;                                       1f2b1j1
   %return%                                                  1f2b1k
```

```
        RETURN (pcrh);                                          1f2b1k1

        END,%+PDP10%                                            1f2b1k2

    %+PDP10% (delpr) %delete local processor%
    PROCEDURE (pcrh, okifldr %-> cost%);                        1f2b2

      %declarations%                                            1f2b2a

        LOCAL                                                   1f2b2a1

          sph, port, nxtpcrh, fkh, nxtcost, evh, cost=0,
          outcome=FALSE;                                        1f2b2a1a

        LOCAL su POINTER, pr POINTER;                           1f2b2a2

      %delete one processor%                                    1f2b2b

        IF pcrh THEN                                            1f2b2b1

          BEGIN                                                 1f2b2b1a

          %locate processor record%                             1f2b2b1b

            pr _ fndrec (vprfld, pcrh, cshare);                 1f2b2b1b1

            INVOKE (prtndl,, pr);                               1f2b2b1b2

            IF (pr.prspldr AND NOT okifldr)                     1f2b2b1b3

            OR pcrh = vcpcrh OR pr.prsecond THEN                1f2b2b1b4

              sigerr (egmhpr);                                  1f2b2b1b4a

          %locate subprocess cost%                              1f2b2b1c

            su _ fndrec (vsufld, sph _ pr.prsph, cshare);       1f2b2b1c1

            INVOKE (palwkp,, su);                               1f2b2b1c2

          %solicit and await signout of processor%              1f2b2b1d

            vjusr (sph, pcrh, cprso);                           1f2b2b1d1

            waicok (pr.prsiecb);                                1f2b2b1d2

            chgrec (pr, cdelete);                               1f2b2b1d3
```

```
            %flush processor from system%                          1f2b2b1e
               prflsh (pcrh);                                      1f2b2b1e1
            %delete JUSR event%                                    1f2b2b1f
               IF evh _ pr,prjuevh THEN delev (evh);               1f2b2b1f1
            %halt processor fork%                                  1f2b2b1g
               oendfk (fkh _ pr,prfkh);                            1f2b2b1g1
            %disconnect addr space from subprocess leader's%       1f2b2b1h
               IF NOT pr,prspldr THEN                              1f2b2b1h1
                  odcnfk (fkh, su,supg, su,supgcnt);               1f2b2b1h1a
            %delete processor fork%                                1f2b2b1i
               odelfk (fkh);                                       1f2b2b1i1
            %decrement processor count%                            1f2b2b1j
               BUMP DOWN su,sucrnpr;                               1f2b2b1j1
               su,sucost _ su,sucost + (cost _ pr,prcost);         1f2b2b1j2
            END                                                    1f2b2b1k
         %delete all processors within controlling subprocess%     1f2b2c
         ELSE                                                      1f2b2c1
            BEGIN                                                  1f2b2c1a
            OPENPORT runfld (vprfld, cdelete : [port]);            1f2b2c1b
            WHILE pr _ PCALL [port] (cprevue : nxtpcrh) DO         1f2b2c1c
               IF pr,prsph = vcsph                                 1f2b2c1c1
               AND NOT pr,prspldr                                  1f2b2c1c2
               AND nxtpcrh # vcpcrh                                1f2b2c1c3
               AND NOT pr,prsecond                                 1f2b2c1c4
               AND PCALL [port] (clock) THEN                       1f2b2c1c5
```

```
                BEGIN                                          1f2b2c1c5a
                nxtcost _                                      1f2b2c1c5b
                   delpr (nxtpcrh, FALSE ;pfail,,
                   $outcome);                                  1f2b2c1c5b1
                IF outcome := TRUE THEN cost _ cost +
                nxtcost;                                       1f2b2c1c5c
                END;                                           1f2b2c1c5d
             END;                                              1f2b2c1d
          %return%                                             1f2b2d
             RETURN (cost);                                    1f2b2d1
             END,%+PDP10%
                                                               1f2b2d2
       %+PDP10% (sipr)  %sign in local processor%
       PROCEDURE (psname REF, pknames REF, shdpgs, psichn, flags
       %-> stupinfo REF, juevh, oflags%);                      1f2b3
          %declarations%                                       1f2b3a
             LOCAL                                             1f2b3a1
                autopcr, seqpr, autordy, encap, i, juevh,
                oflags, len, btmpg, toppg, fkh, pcrh;          1f2b3a1a
             LOCAL hashs REF;                                  1f2b3a2
             LOCAL su POINTER, pr POINTER;                     1f2b3a3
          %catchphrases%                                       1f2b3b
             (pdelfk) CATCHPHRASE; IF abr THEN odelfk (fkh);   1f2b3b1
             (psepfk) CATCHPHRASE; IF abr THEN                 1f2b3b2
                sepfk (pcrh, TRUE);                            1f2b3b2a
             (pendfk) CATCHPHRASE; IF abr THEN oendfk (fkh);   1f2b3b3
          %isolate flags%                                      1f2b3c
             autopcr _ IF flags .A 4B11 THEN 1 ELSE 0;         1f2b3c1
```

```
    seqpr   _ IF flags ,A 2B11 THEN 1 ELSE 0;                1f2b3c2
    autordy _ IF flags ,A 1B11 OR seqpr THEN 1 ELSE 0;       1f2b3c3
    vsplok  _    flags ,A 4B10;                              1f2b3c4
    encap   _ IF flags ,A 2B10 THEN 1 ELSE 0;                1f2b3c5
  %locate subprocess record%                                 1f2b3d
    su _ fndrec (vsufld, vcsph, cshare);                     1f2b3d1
    INVOKE (palwkp,, su);                                    1f2b3d2
  %locate processor record%                                  1f2b3e
    pr _ fndrec (vprfld, vcpcrh, cshare);                    1f2b3e1
    INVOKE (palwkp,, pr);                                    1f2b3e2
  %prevent duplicate signin%                                 1f2b3f
    IF pr,prsignin THEN sigerr (erdsin);                     1f2b3f1
  %save signin parameters%                                   1f2b3g
    pr,prseqpr   _ seqpr;                                    1f2b3g1
    pr,prautordy _ autordy;                                  1f2b3g2
    IF pr,prspldr THEN                                       1f2b3g3
      BEGIN                                                  1f2b3g3a
      %save package list%                                    1f2b3g3b
        IF &pknames THEN                                     1f2b3g3b1
          BEGIN                                              1f2b3g3b1a
          su,supknames _ savent (clist, &pknames);
                                                             1f2b3g3b1b
          su,supkhashs _ &hashs _                            1f2b3g3b1c
            aloent (clist, len _ pknames,L);                 1f2b3g3b1c1
          FOR i _ 1 UP UNTIL > len DO                        1f2b3g3b1d
            #hashs# !_ hash (ELEM #pknames# [i]);
                                                             1f2b3g3b1d1
```

```
            su,supkuses _ aloent (clist, len);      1f2b3g3b1e
            END;                                    1f2b3g3b1f
         %save miscellaneous parameters%            1f2b3g3c
            DIV shdpgs / 1B3, btmpg, toppg;         1f2b3g3c1
            su,supg    _ btmpg;                     1f2b3g3c1a
            su,supgcnt _ toppg - btmpg + 1;         1f2b3g3c1b
            su,suautopcr _ autopcr;                 1f2b3g3c2
            IF su,supsldr THEN *vpsname* _ *psname*; 1f2b3g3c3
         %encapsulate subprocess%                   1f2b3g3d
            IF encap THEN                           1f2b3g3d1
               BEGIN                                1f2b3g3d1a
               %create encapsulator%                1f2b3g3d1b
                  fkh _ ocrtfk ($dencap);           1f2b3g3d1b1
                  INVOKE (pdelfk);                  1f2b3g3d1b2
               %make it an associate processor%     1f2b3g3d1c
                  pcrh _ itdfk (fkh, 0, TRUE);      1f2b3g3d1c1
                  INVOKE (psepfk);                  1f2b3g3d1c2
               %start it%                           1f2b3g3d1d
                  obgnfk (fkh);                     1f2b3g3d1d1
                  INVOKE (pendfk);                  1f2b3g3d1d2
                  su,suepcrh _ pcrh;                1f2b3g3d1d3
                  su,suefkh  _ fkh;                 1f2b3g3d1d4
               END;                                 1f2b3g3d1e
         END;                                       1f2b3g3e
      %create JUSR event%                           1f2b3h
```

```
    IF seqpr THEN                                          1f2b3h1
      BEGIN                                                1f2b3h1a
      pr.prabrchn _ psichn;                                1f2b3h1b
      juevh _ 0;                                           1f2b3h1c
      END                                                  1f2b3h1d
    ELSE                                                   1f2b3h2
      pr.prjuevh _ juevh _                                 1f2b3h2a
        crtev (cprocessor, cjuevlen, psichn);              1f2b3h2a1
%encapsulate processor%                                    1f2b3i
    IF fkh _ su.suefkh THEN                                1f2b3i1
      BEGIN                                                1f2b3i1a
      omovfk (pr.prfkh, fkh);                              1f2b3i1b
      pr.prsuper _ su.suepcrh;                             1f2b3i1c
      END;                                                 1f2b3i1d
%notify system of signin%                                  1f2b3j
    pr.prsignin _ TRUE;                                    1f2b3j1
    sigecb (pr.prsiecb, cok);                              1f2b3j2
%construct flags%                                          1f2b3k
    oflags _                                               1f2b3k1
      vtr1dr*4B11 + su.sups1dr*2B11 + pr.prsp1dr*1B11;
                                                           1f2b3k1a
%ready processor%                                          1f2b3l
    IF autordy THEN rdypr ();                              1f2b3l1
%return%                                                   1f2b3m
    RETURN (savent (cblock, pr.prstup), juevh, oflags);
                                                           1f2b3m1
```

```
        END,%+PDP10%                                                    1f2b3m2

    %+PDP10% (sopr)  %sign out local processor%
    PROCEDURE;                                                          1f2b4
      %declarations%                                                    1f2b4a
        LOCAL pr POINTER;                                               1f2b4a1
      %locate processor record%                                         1f2b4b
        pr _ fndrec (vprfld, vcpcrh, cshare);                           1f2b4b1
        INVOKE (palwkp,, pr);                                           1f2b4b2
      %prevent duplicate signout%                                       1f2b4c
        IF NOT pr.prsignin := FALSE THEN sigerr (erisin);               1f2b4c1
      %notify system of signout%                                        1f2b4d
        sigecb (pr,prsiecb, cok);                                       1f2b4d1
      %return%                                                          1f2b4e
        RETURN;                                                         1f2b4e1
        END,%+PDP10%                                                    1f2b4e2

    %+PDP10% (rdypr) %ready local processor for service
    request assignment%
    PROCEDURE;                                                          1f2b5
      %declarations%                                                    1f2b5a
        LOCAL ecb;                                                      1f2b5a1
        LOCAL su POINTER, pr POINTER;                                   1f2b5a2
      %locate subprocess record%                                        1f2b5b
        su _ fndrec (vsufld, vcsph, cshare);                            1f2b5b1
        INVOKE (palwkp,, su);                                           1f2b5b2
      %locate processor record%                                         1f2b5c
```

```
        pr _ fndrec (vprfld, vcpcrh, cshare);                    1f2b5c1
        INVOKE (palwkp,, pr);                                    1f2b5c2
    %bump ready count%                                           1f2b5d
        BUMP pr.prrycnt;                                         1f2b5d1
    %notify system of processor's availability%                  1f2b5e
        WHILE ecb _ otest (su.suwtecbs, TRUE) DO                 1f2b5e1
            sigecb (ecb, cok);                                   1f2b5e1a
    %return%                                                     1f2b5f
        RETURN;                                                  1f2b5f1
        END,%+PDP10%
                                                                 1f2b5f2

%+PDP10% (itdfk) %introduce fork to DPS%
PROCEDURE (fkh, stupinfo REF, self %-> pcrh%);                   1f2b6
    %declarations%                                               1f2b6a
        LOCAL pcrh;                                              1f2b6a1
        LOCAL cpr POINTER, pr POINTER;                           1f2b6a2
    %locate controlling processor record%                        1f2b6b
        cpr _ fndrec (vprfld, vcpcrh, cshare);                   1f2b6b1
        INVOKE (palwkp,, cpr);                                   1f2b6b2
    %create processor record%                                    1f2b6c
        pr _ crtrec (vprfld, csubprocess : pcrh);                1f2b6c1
        INVOKE (prtnkp,, pr);                                    1f2b6c2
        pr.prsuper  _ vcpcrh;                                    1f2b6c3
        pr.prsecond _ TRUE;                                      1f2b6c4
        pr.prstup   _ savent (cblock, &stupinfo);                1f2b6c5
        pr.prsph    _ vcsph;                                     1f2b6c6
```

```
        %fetch handle for fork%                                    1f2b6d
           pr,prfkh _ IF self                                      1f2b6d1
              THEN fkh                                             1f2b6d1a
              ELSE oitdfk (cpr,prfkh, fkh);                        1f2b6d1b
        %intercept DPS operations%                                 1f2b6e
           otrpfk (fkh);                                           1f2b6e1
        %return%                                                   1f2b6f
           RETURN (pcrh);                                          1f2b6f1
           END,%+PDP10%
                                                                   1f2b6f2
     %+PDP10% (sepfk) %separate fork from DPS%
     PROCEDURE (pcrh, self);                                       1f2b7
        %declarations%                                             1f2b7a
           LOCAL port, evh, nxtpcrh, fkh;                          1f2b7a1
           LOCAL pr POINTER;                                       1f2b7a2
        %separate one fork%                                        1f2b7b
           IF pcrh THEN                                            1f2b7b1
              BEGIN                                                1f2b7b1a
              %locate processor record%                            1f2b7b1b
                 pr _ fndrec (vprfld, pcrh, cshare);               1f2b7b1b1
                 INVOKE (prtndl,, pr);                             1f2b7b1b2
                 IF NOT (pr,prsecond AND pr,prsuper = vcpcrh)
                 THEN                                              1f2b7b1b3
                    sigerr (egmhpr);                               1f2b7b1b3a
              %solicit and await signout of processor%             1f2b7b1c
                 vjusr (pr,prsph, pcrh, cprso);                    1f2b7b1c1
```

```
                waicok (pr,prsiecb);                                    1f2b7b1c2
                chgrec (pr, cdelete);                                   1f2b7b1c3
             %release DPS operations%                                   1f2b7b1d
                outrfk (fkh _ pr,prfkh);                                1f2b7b1d1
             %release fork handle%                                      1f2b7b1e
                IF NOT self THEN osepfk (fkh);                          1f2b7b1e1
             %flush processor from system%                              1f2b7b1f
                prflsh (pcrh);                                          1f2b7b1f1
             %delete JUSR event%                                        1f2b7b1g
                IF evh _ pr.prjuevh THEN delev (evh);                   1f2b7b1g1
             END                                                        1f2b7b1h
          %separate all forks introduced by controlling
          processor%                                                    1f2b7c
            ELSE                                                        1f2b7c1
              BEGIN                                                     1f2b7c1a
              OPENPORT runfld (vprfld, cshare : [port]);                1f2b7c1b
              WHILE pr _ PCALL [port] (cprevue : nxtpcrh) DO            1f2b7c1c
                IF pr.prsecond AND pr.prsuper = vcpcrh                  1f2b7c1c1
                AND PCALL [port] (clock) THEN                           1f2b7c1c2
                   sepfk (nxtpcrh ;pignore);                            1f2b7c1c2a
              END;                                                      1f2b7c1d
          %return%                                                      1f2b7d
            RETURN;                                                     1f2b7d1
            END.%+PDP10%
                                                                        1f2b7d2
%channels%                                                              1f2c
```

```
    %+PDP10% (sndch) %send data on local channel%
    PROCEDURE (poh, block REF);                                  1f2c1

       %send data on channel%                                    1f2c1a

          isndch (poh, &block);                                  1f2c1a1

       %return%                                                  1f2c1b

          RETURN;                                                1f2c1b1

          END,%+PDP10%                                           1f2c1b2

    %+PDP10% (rcvch) %receive data on local channel%
    PROCEDURE (poh %-> block REF%);                              1f2c2

       %return%                                                  1f2c2a

          RETURN (ircvch (poh));                                 1f2c2a1

          END,%+PDP10%                                           1f2c2a2

%locks%                                                          1f2d

    %+PDP10% (crtlk) %create local lock%
    PROCEDURE (scope %-> lkh%);                                  1f2d1

       %declarations%                                            1f2d1a

          LOCAL lkh;                                             1f2d1a1

       %create lock record%                                      1f2d1b

          unlckrec (crtrec (vlkfld, scope : lkh), ckeep);        1f2d1b1

       %return%                                                  1f2d1c

          RETURN (lkh);                                          1f2d1c1

          END,%+PDP10%                                           1f2d1c2

    %+PDP10% (dellk) %delete local lock%
    PROCEDURE (lkh);                                             1f2d2

       %declarations%                                            1f2d2a
```

```
        LOCAL port, nxtlkh;                                    1f2d2a1
        LOCAL lk POINTER;                                      1f2d2a2
      %delete one lock%                                        1f2d2b
        IF lkh THEN                                            1f2d2b1
          BEGIN                                                1f2d2b1a
          %locate lock record%                                 1f2d2b1b
            lk _ fndrec (vlkfld, lkh, cdelete);                1f2d2b1b1
            INVOKE (prtndl,, lk);                              1f2d2b1b2
          %terminate LCB%                                      1f2d2b1c
            trmlcb (lk,lklcb);                                 1f2d2b1c1
          END                                                  1f2d2b1d
      %delete all locks%                                       1f2d2c
        ELSE                                                   1f2d2c1
          BEGIN                                                1f2d2c1a
          OPENPORT runfld (vlkfld, cdelete ; [port]);          1f2d2c1b
          WHILE lk _ PCALL [port] (clock ; nxtlkh) DO          1f2d2c1c
            dellk (nxtlkh ;ignore);                            1f2d2c1c1
          END;                                                 1f2d2c1d
      %return%                                                 1f2d2d
        RETURN;                                                1f2d2d1
        END,%+PDP10%
                                                               1f2d2d2
%+PDP10% (setlk) %set local lock%
PROCEDURE (lkh, type, scope, flags %-> lsh%);                  1f2d3
    %declarations%                                             1f2d3a
        LOCAL ldsc, wait, ecb;                                 1f2d3a1
```

```
        LOCAL lckid [cmlkidl];                                  1f2d3a2
        LOCAL lk POINTER;                                       1f2d3a3
    %isolate flags%                                             1f2d3b
        wait _ NOT (flags ,A 4B11);                             1f2d3b1
    %locate lock record%                                        1f2d3c
        lk _ fndrec (vlkfld, lkh, cshare);                      1f2d3c1
        INVOKE (palwkp,, lk);                                   1f2d3c2
    %allocate ECB%                                              1f2d3d
        ecb _ IF wait THEN aloecb (1) ELSE 0;                   1f2d3d1
        INVOKE (palwrb,, ecb);                                  1f2d3d2
    %set lock%                                                  1f2d3e
        lckid     _ lk,lklcb;                                   1f2d3e1
        lckid [1] _ type;                                       1f2d3e2
        lckid [2] _ scope;                                      1f2d3e3
        ldsc _ setlcb ($lckid, ecb, TRUE);                      1f2d3e4
        waicok (ecb);                                           1f2d3e5
    %return%                                                    1f2d3f
        RETURN (ldsc,ldlsh);                                    1f2d3f1
        END,%+PDP10%
                                                                1f2d3f2
%+PDP10% (remlk) %remove local lock%
PROCEDURE (lkh, lsh);                                           1f2d4
    %declarations%                                              1f2d4a
        LOCAL ldsc=0;                                           1f2d4a1
        LOCAL lk POINTER;                                       1f2d4a2
    %locate lock record%                                        1f2d4b
```

```
            lk _ fndrec (vlkfld, lkh, cshare);                    1f2d4b1
            INVOKE (palwkp,, lk);                                 1f2d4b2
         %remove lock%                                            1f2d4c
            ldsc,ldlcb _ lk,lklcb;                                1f2d4c1
            ldsc,ldlsh _ lsh;                                     1f2d4c2
            remlcb (ldsc);                                        1f2d4c3
         %return%                                                 1f2d4d
            RETURN;                                               1f2d4d1
            END,%+PDP10%
                                                                  1f2d4d2
      %events%                                                    1f2e
      (crtev) %create local event%
      PROCEDURE (scope, length, psichan %-> evh%);                1f2e1
         %declarations%                                           1f2e1a
            LOCAL evh;                                            1f2e1a1
            LOCAL ev POINTER;                                     1f2e1a2
         %create event record%                                    1f2e1b
            ev _ crtrec (vevfld, scope : evh);                    1f2e1b1
            INVOKE (prtnkp,, ev);                                 1f2e1b2
            ev,evchan _ psichan;                                  1f2e1b3
         %allocate ECB%                                           1f2e1c
            IF NOT (length IN [1, cmevlen]) THEN                  1f2e1c1
               sigerr (evilen);                                   1f2e1c1a
            ev,evecb _ aloecb (length);                           1f2e1c2
         %return%                                                 1f2e1d
```

```
        RETURN (evh);                                              1f2e1d1
        END.
                                                                   1f2e1d2
(delev) %delete local event%
PROCEDURE (evh);                                                   1f2e2
   %declarations%                                                  1f2e2a
      LOCAL port, nxtevh;                                          1f2e2a1
      LOCAL ev POINTER;                                            1f2e2a2
   %delete one event%                                              1f2e2b
      IF evh THEN                                                  1f2e2b1
         BEGIN                                                     1f2e2b1a
         %locate event record%                                     1f2e2b1b
            ev _ fndrec (vevfld, evh, cdelete);                    1f2e2b1b1
            INVOKE (prtndl,, ev);                                  1f2e2b1b2
         %flush event from system%                                 1f2e2b1c
            evflsh (evh);                                          1f2e2b1c1
         END                                                       1f2e2b1d
   %delete all events%                                             1f2e2c
      ELSE                                                         1f2e2c1
         BEGIN                                                     1f2e2c1a
         OPENPORT runfld (vevfld, cdelete ; [port]);               1f2e2c1b
         WHILE ev _ PCALL [port] (clock : nxtevh) DO               1f2e2c1c
            delev (nxtevh ;pignore);                               1f2e2c1c1
         END;                                                      1f2e2c1d
   %return%                                                        1f2e2d
      RETURN;                                                      1f2e2d1
```

```
        END,                                                    1f2e2d2

    (sigev) %signal local event%
    PROCEDURE (evh, code, wheelornot);                          1f2e3

      %declarations%                                            1f2e3a

        LOCAL chan;                                             1f2e3a1

        LOCAL ev POINTER, pr POINTER;                           1f2e3a2

      %locate event record%                                     1f2e3b

        vwhlpls _ wheelornot;                                   1f2e3b1

        ev _ fndrec (vevfld, evh, csignal);                     1f2e3b2

        INVOKE (palwkp,, ev);                                   1f2e3b3

      %signal event%                                            1f2e3c

        IF (chan _ ev.evchan) <= cmchno THEN                    1f2e3c1

          BEGIN                                                 1f2e3c1a

          pr _ fndrec (vprfld, rdrec (cpcrh, ev), cshare);
                                                                1f2e3c1b
          INVOKE (palwkp,, pr);                                 1f2e3c1c

          osigfk (pr.prfkh, chan);                              1f2e3c1d

          END;                                                  1f2e3c1e

        sigecb (ev,evecb, code);                                1f2e3c2

      %return%                                                  1f2e3d

        RETURN;                                                 1f2e3d1

        END,
                                                                1f2e3d2

    %+PDP10% (tstev) %test for signalled local event%
    PROCEDURE (evh %-> code, newlength%);                       1f2e4

      %declarations%                                            1f2e4a

        LOCAL code, length;                                     1f2e4a1
```

```
        LOCAL ev POINTER;                                          1f2e4a2

    %locate event record%                                          1f2e4b

        ev _ fndrec (vevfld, evh, cwait);                          1f2e4b1

        INVOKE (palwkp,, ev);                                      1f2e4b2

    %test event for completion%                                    1f2e4c

        code _ otest (ev,evecb, TRUE : length);                    1f2e4c1

    %return%                                                       1f2e4d

        RETURN (code, length);                                     1f2e4d1

        END,%+PDP10%                                               1f2e4d2

%+PDP10% (waiev) %wait for signalled local event%
PROCEDURE (evhs REF %=> code, index, length%);                     1f2e5

    %declarations%                                                 1f2e5a

        LOCAL code, i, length, ub;                                 1f2e5a1

        LOCAL LIST evs [cmecbs];                                   1f2e5a2

        LOCAL ecbs REF;                                            1f2e5a3

        LOCAL ev POINTER;                                          1f2e5a4

    %catchphrases%                                                 1f2e5b

        (pnulst)    CATCHPHRASE; IF abrtn () THEN
        NULL-LISTS;                                                1f2e5b1

        (punlckevs) CATCHPHRASE; IF abrtn () THEN                  1f2e5b2

            BEGIN                                                  1f2e5b2a

            ub _ evs,L;                                            1f2e5b2b

            FOR i _ 1 UP UNTIL > ub DO                             1f2e5b2c

                unlckrec (ELEM #evs# [i], cKeep);                  1f2e5b2c1

            END;                                                   1f2e5b2d
```

```
            %assure local lists released%                                   1f2e5c
               INVOKE (pnulst);                                             1f2e5c1
            %allocate ECB addr list%                                        1f2e5d
               &ecbs _ aloent (clist, ub _ evhs,L);                         1f2e5d1
               INVOKE (palwrl,, &ecbs);                                     1f2e5d2
            %construct list of ECB addrs%                                   1f2e5e
               INVOKE (punlckevs);                                          1f2e5e1
               FOR i _ 1 UP UNTIL > ub DO                                   1f2e5e2
                  BEGIN                                                     1f2e5e2a
                  #evs# !_ ev _                                             1f2e5e2b
                     fndrec (vevfld, ELEM #evhs# [i], cwait);               1f2e5e2b1
                  #ecbs# !_ ev.evecb;                                       1f2e5e2c
                  END;                                                      1f2e5e2d
            %wait for event%                                                1f2e5f
               code _ owait (&ecbs, TRUE : i, length);                      1f2e5f1
            %return%                                                        1f2e5g
               RETURN (code, i, length);                                    1f2e5g1
               END,%+PDP10%
                                                                            1f2e5g2
         %timers%                                                           1f2f
            %+PDP10% (setmr) %set virtual interval timer%
            PROCEDURE (interval, ecb REF, evh, scope %-> tmh%);             1f2f1
               %declarations%                                               1f2f1a
                  LOCAL tmh, left;                                          1f2f1a1
                  LOCAL tm POINTER;                                         1f2f1a2
               %create timer record%                                        1f2f1b
```

```
        tm _                                                    1f2f1b1

           crtrec (vtmfld, IF &ecb THEN call ELSE scope :
           tmh);                                                1f2f1b1a

        INVOKE (prtnkp,, tm);                                   1f2f1b2

        tm.tminterval  _ interval;                              1f2f1b3

        tm.tmleft      _ interval + otsttmr (FALSE : left);
                                                                1f2f1b4
        IF &ecb THEN tm.tmecb _ &ecb                            1f2f1b5

        ELSE tm.tmevh _ evh;                                    1f2f1b6

     %reset system timer if necessary%                          1f2f1c

        IF left > interval THEN updtmr ();                      1f2f1c1

     %return%                                                   1f2f1d

        RETURN (tmh);                                           1f2f1d1

        END.%+PDP10%
                                                                1f2f1d2

%+PDP10% (tstmr) %test/cancel virtual interval timer%
PROCEDURE (tmh, flags %-> gone, left%);                         1f2f2

   %declarations%                                               1f2f2a

      LOCAL gone, left, cancel;                                 1f2f2a1

      LOCAL tm POINTER;                                         1f2f2a2

   %isolate flags%                                              1f2f2b

      cancel _ flags .A 4B11;                                   1f2f2b1

   %locate timer record%                                        1f2f2c

      tm _ fndrec (vtmfld, tmh, cdelete);                       1f2f2c1

      INVOKE (palwkp,, tm);                                     1f2f2c2

   %note gone and left%                                         1f2f2d

      left _ tm.tmleft - otsttmr (FALSE);                       1f2f2d1
```

```
                gone _ tm,tminterval - left;                    1f2f2d2
              %cancel system timer if necessary%                1f2f2e
                IF cancel THEN                                  1f2f2e1
                  BEGIN                                         1f2f2e1a
                  delrec (tm);                                  1f2f2e1b
                  updtmr ();                                    1f2f2e1c
                  END;                                          1f2f2e1d
              %return%                                          1f2f2f
                RETURN (gone, left);                            1f2f2f1
                END,%+PDP10%
                                                                1f2f2f2
    %debugging%                                                 1f3

      (setrc) %set remote DPS trace word%
      PROCEDURE (ph, setting);                                  1f3a
        %set local trace word%                                  1f3a1
          IF ph = cfph THEN dtrace _ setting                    1f3a1a
        %set remote trace word%                                 1f3a2
          ELSE syscal (ph, cxsetrc, setting);                   1f3a2a
        %return%                                                1f3a3
          RETURN;                                               1f3a3a
          END,
                                                                1f3a3b
%messages%                                                      1g

    (xcalpe) %call local procedure%
    PROCEDURE (ch, pkh, pname REF, argl REF, arglmsk REF, reslmsk
    REF, priority);                                             1g1

      %declarations%                                            1g1a
```

```
    LOCAL                                                          1g1a1

       number, intpname, resc, pcrh, flags, cost=0,
       outcome=csuccess;                                           1g1a1a

    LOCAL LIST msg [cmmsgl];                                       1g1a2

    LOCAL resl REF =0;                                             1g1a3

    LOCAL ca POINTER, pk POINTER, entry POINTER;                   1g1a4

%catchphrases%                                                     1g1b

    (pabrpe) CATCHPHRASE (: vdgmsg, pcrh); CASE TRUE OF            1g1b1

      = abr ():                                                    1g1b1a

        BEGIN                                                      1g1b1a1

        outcome _ caborted;                                        1g1b1a2

        relent (clist, &resl := 0);                                1g1b1a3

        &resl _ aloent (clist, 2);                                 1g1b1a4

        #resl# !_                                                  1g1b1a5

          USE descrb (encstruc (cindex, SIGNAL)),                  1g1b1a5a

          USE descrb (encstruc (ccharstr, vdgmsg));                1g1b1a5b

        DROP (palwrb);                                             1g1b1a6

        DROP (palwkp);                                             1g1b1a7

        DROP (palwdl);                                             1g1b1a8

        TERMINATE;                                                 1g1b1a9

        END;                                                       1g1b1a10

      = rtn ():                                                    1g1b1b

        BEGIN                                                      1g1b1b1

        #msg# _ 0, cmrtnpe, ch, &resl, 0, 0, outcome, cost;
                                                                   1g1b1b2

        sndps (vcph, $msg);                                        1g1b1b3
```

```
            relent (clist, &resl);                                  1g1b1b4
            NULL=LISTS;                                             1g1b1b5
            END;                                                    1g1b1b6
         = (SIGNALTYPE = notetype AND SIGNAL = cselpr);             1g1b1c
            BEGIN                                                   1g1b1c1
            ca.capcrh  _ pcrh;                                      1g1b1c2
            ca.caseqpr _ [vdgmsg].prseqpr;                          1g1b1c3
            END;                                                    1g1b1c4
         ENDCASE;                                                   1g1b1d
      %route outcome to invoking process%                           1g1c
         INVOKE (pabrpe, RETURN);                                   1g1c1
      %create call record%                                          1g1d
         ca _ crtrec (vcafld, call : vcch);                         1g1d1
         INVOKE (palwdl,, ca);                                      1g1d2
         ca.castate _ crunloc;                                      1g1d3
         ca.cacch   _ ch;                                           1g1d4
         ca.caph    _ vcph;                                         1g1d5
         ca.capkh   _ pkh;                                          1g1d6
         ca.capname _ savent (cblock, &pname);                      1g1d7
      %mask argument list%                                          1g1e
         mskarlst (carglist, &argl, &arglmsk);                      1g1e1
         ca.camskl _ savent (clist, &reslmsk);                      1g1e2
      %user procedure%                                              1g1f
         IF pkh THEN                                                1g1f1
            BEGIN                                                   1g1f1a
```

```
        %locate package record%                                    1g1f1b
           pk _ fndrec (vpkfld, pkh, cshare);                      1g1f1b1
           INVOKE (palwkp,, pk);                                   1g1f1b2
        %verify package alive%                                     1g1f1c
           IF pk,pkdead THEN sigerr (ekfded);                      1g1f1c1
        %decode procedure name%                                    1g1f1d
           intpname _ decstruc (cucstr, &pname, 0);                1g1f1d1
           INVOKE (palwrb,, intpname);                             1g1f1d2
        %call local procedure%                                     1g1f1e
           chgrec (ca, cshare);                                    1g1f1e1
           cost _                                                  1g1f1e2
              vjusr (ca,casph _ pk,pksph, 0, priority*1B6 +
              cpecal, vcch, pk,pkintpkh, intpname, &argl :
              flags, &res1);                                       1g1f1e2a
           outcome _ IF flags .A 4B11                              1g1f1e3
              THEN cfailure                                        1g1f1e3a
              ELSE csuccess;                                       1g1f1e3b
           chgrec (ca, cinternal);                                 1g1f1e4
        %note cost%                                                1g1f1f
           pk,pkcost _ pk,pkcost + cost;                           1g1f1f1
        END                                                        1g1f1g
     %system procedure%                                            1g1g
        ELSE                                                       1g1g1
           BEGIN                                                   1g1g1a
           %lookup procedure%                                      1g1g1b
              number _ decstruc (cindex, &pname, 0);               1g1g1b1
```

```
            entry _ fndsyp (number);                                    1g1g1b2
            IF NOT (vsignin OR number = cxinips) THEN                   1g1g1b3
               sigerr (erisin);                                         1g1g1b3a
         %decode arguments%                                             1g1g1c
            decprms (&argl, entry,enargc, entry+2);                     1g1g1c1
         %allocate result list%                                         1g1g1d
            &resl _ aloent (clist, resc _ entry.enresc);                1g1g1d1
         %call system procedure%                                        1g1g1e
            cost _ ocost (0);                                           1g1g1e1
            l10cal (entry, &argl, &resl);                               1g1g1e2
            vxcost _ vxcost + (cost _ ocost (0) - cost);                1g1g1e3
         %encode results%                                               1g1g1f
            encprms (&resl, resc, entry+4, 0);                          1g1g1f1
         END;                                                           1g1g1g
      %mask result list%                                                1g1h
         mskarlst (creslist, &resl, ca,camskl);                         1g1h1
      %return%                                                          1g1i
         RETURN;                                                        1g1i1
         END.
                                                                        1g1i2

(xrecpe) %recall local procedure%
PROCEDURE (ch, argl REF, arglmsk REF, reslmsk REF);                     1g2
      %declarations%                                                    1g2a
         LOCAL acqevh, code, locch;                                     1g2a1
         LOCAL ca POINTER;                                              1g2a2
      %locate call record%                                              1g2b
```

```
        ca _ fndcall (clocal, ch, cexternal : locch);                    1g2b1
        INVOKE (palwkp,, ca);                                            1g2b2
    %verify call state%                                                  1g2c
        IF ca.castate # crunrem THEN sigerr (epixrc);                    1g2c1
    %mask and save argument list%                                        1g2d
        mskarlst (                                                       1g2d1
            carglist, ca.caprml _ savent (clist, &argl),
            &arglmsk);                                                   1g2d1a
    %save result list mask%                                              1g2e
        ca.camskl _ savent (clist, &reslmsk);                            1g2e1
    %notify system and local procedure%                                  1g2f
        ca.castate _ creturned;                                          1g2f1
        sigecb (ca.caqecb, cvisit);                                      1g2f2
        IF acqevh _ ca.caqevh THEN                                       1g2f3
            BEGIN                                                        1g2f3a
            code.LH _ locch; code.RH _ cvisit;                           1g2f3b
            sigev (acqevh, code, TRUE);                                  1g2f3c
            END;                                                         1g2f3d
    %return%                                                             1g2g
        RETURN;                                                          1g2g1
        END.                                                             1g2g2

(xrtnpe) %return to local caller%
PROCEDURE (ch, res1 REF, reslmsk REF, arglmsk REF, outcome,
cost);                                                                   1g3
    %declarations%                                                       1g3a
        LOCAL acqevh, code;                                              1g3a1
```

```
    LOCAL ca POINTER;                                          1g3a2
%locate call record%                                           1g3b
    ca _ fndcall (cremote, ch, cexternal);                     1g3b1
    INVOKE (palwkp,, ca);                                      1g3b2
%verify call state%                                            1g3c
    IF ca.castate # crunrem THEN sigerr (epixrn);              1g3c1
%mask and save result list%                                    1g3d
    mskarlst (                                                 1g3d1
        carglist, ca.caprml _ savent (clist, &resl),
        &reslmsk);                                             1g3d1a
%save argument list mask / cost%                               1g3e
    ca.camskl _ savent (clist, &arglmsk);                      1g3e1
    ca.cacost _ cost;                                          1g3e2
%notify system and local procedure%                            1g3f
    ca.castate _ creturned;                                    1g3f1
    sigecb (ca,caqecb, outcome);                               1g3f2
    IF acqevh _ ca.caqevh THEN                                 1g3f3
        BEGIN                                                  1g3f3a
        code.LH _ ch; code.RH _ outcome;                       1g3f3b
        sigev (acqevh, code, TRUE);                            1g3f3c
        END;                                                   1g3f3d
%return%                                                       1g3g
    RETURN;                                                    1g3g1
    END.
                                                               1g3g2
```

```
    (xerror) %error%
    PROCEDURE (number, msg REF);                                    1g4

      %return%                                                      1g4a

        RETURN;                                                     1g4a1

        END,                                                        1g4a2

  %system procedures%                                               1h

    %processes%                                                     1h1

      (xinips) %initialize process%
      PROCEDURE (intrahostaddr REF, userinfo REF, stupinfo REF,
      spsdesc REF, pkinfo REF, pkhs REF %-> fpsdesc REF, pkhs
      REF%);                                                        1h1a

        %declarations%                                              1h1a1

          LOCAL LIST fpsdesc [cmpdcl], psloc [cmplocl];             1h1a1a

          LOCAL ps POINTER;                                         1h1a1b

        %catchphrases%                                              1h1a2

          (pdelsp) CATCHPHRASE; IF abr () THEN delsp (vpsldr,
          TRUE);                                                    1h1a2a

        %complete superior's process record%                        1h1a3

          IF NOT vtrldr THEN                                        1h1a3a

            BEGIN                                                   1h1a3a1

            %verify caller%                                         1h1a3a2

              IF vcph # vsph THEN sigerr (esumsg);                  1h1a3a2a

            %locate superior's process record%                      1h1a3a3

              ps _ fndrec (vpsfld, vsph, cshare);                   1h1a3a3a

              INVOKE (palwkp,, ps);                                 1h1a3a3b

            %save superior's description%                           1h1a3a4

              ps.pspsdesc _ savent (cblock, &spsdesc);              1h1a3a4a
```

```
            END;                                                    1h1a3a5

        %create process leader subprocess%                          1h1a4

            vpsldr _                                                1h1a4a

                crtsp (&intrahostaddr, &stupinfo, call, cdfprio,
                TRUE);                                              1h1a4a1

            INVOKE (pdelsp);                                        1h1a4b

        %construct self's description%                              1h1a5

            ps _ fndrec (vpsfld, vfph, cshare);                     1h1a5a

            INVOKE (palwkp,, ps);                                   1h1a5b

            #psloc# _ vfhost, vfjob, 1;                             1h1a5c

            #fpsdesc# _                                             1h1a5d

                $vpsname, &intrahostaddr, $psloc,                   1h1a5d1

                IF &userinfo THEN ELEM #userinfo# [1] ELSE 0;       1h1a5d2

            ps,pspsdesc _ vfpsdesc _ encstruc (cpsdesc, $fpsdesc);  1h1a5e

        %open packages%                                             1h1a6

            IF &pkinfo THEN                                         1h1a6a

                xopnpk (ELEM #pkinfo# [1], ELEM #pkinfo# [2], ELEM
                #pkinfo# [3], &pkhs);                               1h1a6a1

        %return%                                                    1h1a7

            vsignin _ TRUE;                                         1h1a7a

            RETURN (savent (cblock, vfpsdesc), &pkhs);              1h1a7b

            END,
                                                                    1h1a7c

    %+PDP10% (xtrmps) %terminate process%
    PROCEDURE %(=> cost)%;                                          1h1b

        %declarations%                                              1h1b1

            LOCAL cost;                                             1h1b1a
```

```
    %verify source of message%                                    1h1b2
       IF vcph # vsph THEN sigerr (esumsg);                       1h1b2a
    %drain system%                                                1h1b3
       vdrain _ vwheel _ TRUE;                                    1h1b3a
    %delete all subprocesses%                                     1h1b4
       cost _ delsp (0, TRUE);                                    1h1b4a
    %delete all inferior processes%                               1h1b5
       delps (0);                                                 1h1b5a
    %return%                                                      1h1b6
       RETURN (cost);                                             1h1b6a
       END,%+PDP10%                                               1h1b6b

(xcrtchh) %create local logical channel half%
PROCEDURE (psdesc REF, stupinfo REF, chainsh, modelsh %->
sh, ph%);                                                         1h1c
    %declarations%                                                1h1c1
       LOCAL nextph, nextsh, returnph, returnsh;                  1h1c1a
       LOCAL ps POINTER, sg POINTER, sgmod POINTER =0;            1h1c1b
    %create necessary record%                                     1h1c2
       CASE modelsh OF                                            1h1c2a
          = 0: %create new process record%                        1h1c2a1
             BEGIN                                                1h1c2a1a
             %create new process record%                          1h1c2a1b
                ps _ crtrec (vpsfld, call : returnph);            1h1c2a1b1
                INVOKE (prtnkp,, ps);                             1h1c2a1b2
                ps,pstype   _ cintroduced;                        1h1c2a1b3
```

```
            ps.pspsdesc _ savent (cblock, &psdesc);     1h1c2a1b4
            ps.psadjph  _ vcph;                         1h1c2a1b5
            ps.psadjsh  _ chainsh;                      1h1c2a1b6
            returnsh _ returnph;                        1h1c2a1b7
         %obtain local process' OK%                     1h1c2a1c
            vjusr (vpsldr, 0, cokips, returnph,
            &stupinfo);                                 1h1c2a1c1
         END;                                           1h1c2a1d
      < csgmnh: %duplicate process record%              1h1c2a2
         BEGIN                                          1h1c2a2a
         %locate model process record%                  1h1c2a2b
            ps _ fndrec (vpsfld, modelsh, cshare);      1h1c2a2b1
            INVOKE (palwkp,, ps);                       1h1c2a2b2
            IF ps.pstype # cintroduced                  1h1c2a2b3
            OR ps.psadjph # vcph THEN sigerr (egmhsg);  1h1c2a2b4
         %create new process record%                    1h1c2a2c
            REPEAT CASE (0);                            1h1c2a2c1
         END;                                           1h1c2a2d
      ENDCASE %duplicate segment record%                1h1c2a3
         BEGIN                                          1h1c2a3a
         %locate model segment record%                  1h1c2a3b
            sgmod _ fndrec (vsgfld, modelsh, cshare);   1h1c2a3b1
            INVOKE (palwkp,, sgmod);                    1h1c2a3b2
         %create new segment record%                    1h1c2a3c
            sg _ crtrec (vsgfld, call : returnsh);      1h1c2a3c1
```

```
                INVOKE (prtnkp,, sg);                                   1h1c2a3c2

                sg,sgph1 _ vcph;                                        1h1c2a3c3

                sg,sgsh1 _ chainsh;                                     1h1c2a3c4

                sg,sgph2 _ nextph _ chndir (sgmod, vcph :
                nextsh);                                                1h1c2a3c5

             %propagate duplication request%                            1h1c2a3d

                syscal                                                  1h1c2a3d1

                   (nextph, cxcrtchh, &psdesc, &stupinfo,
                   returnsh, nextsh : sg,sgsh2, returnph);
                                                                        1h1c2a3d1a
             END;                                                       1h1c2a3e

          %return%                                                      1h1c3

             RETURN (returnsh, returnph);                               1h1c3a

             END,
                                                                        1h1c3b

       (xdelchh) %delete local logical channel half%
       PROCEDURE (sh %-> cost%);                                        1h1d

          %declarations%                                                1h1d1

             LOCAL nextph, nextsh, nextdead, cost;                      1h1d1a

             LOCAL ps POINTER, sg POINTER;                              1h1d1b

          %delete channel end%                                          1h1d2

             IF sh < csgmnh THEN                                        1h1d2a

                BEGIN                                                   1h1d2a1

                %locate process record%                                 1h1d2a2

                   ps _ fndrec (vpsfld, sh, cdelete);                   1h1d2a2a

                   INVOKE (prtndl,, ps);                                1h1d2a2b

                   IF (ps,pstype # cintroduced                          1h1d2a2c

                      OR ps,psadjph # vcph)                             1h1d2a2c1
```

```
                AND NOT vwheel THEN sigerr (egmhsg);                1h1d2a2d
              %obtain local process' OK%                            1h1d2a3
                vjusr (vPsldr, 0, coksPs, sh);                      1h1d2a3a
              %flush process from system%                           1h1d2a4
                psflsh (sh);                                        1h1d2a4a
                cost _ ps.pscost;                                   1h1d2a4b
              END                                                   1h1d2a5
            %delete segment record%                                 1h1d3
          ELSE                                                      1h1d3a
            BEGIN                                                   1h1d3a1
            %locate segment record%                                 1h1d3a2
              sg _ fndrec (vsgfld, sh, cdelete);                    1h1d3a2a
              INVOKE (prtndl,, sg);                                 1h1d3a2b
            %propagate deletion request%                            1h1d3a3
              nextph _ chndir (sg, vcph : nextsh, nextdead);        1h1d3a3a
              IF NOT nextdead THEN                                  1h1d3a3b
                syscal (nextph, cxdelchh, nextsh : cost);           1h1d3a3b1
            END;                                                    1h1d3a4
          %return%                                                  1h1d4
            RETURN (cost);                                          1h1d4a
            END,
                                                                    1h1d4b
%packages%                                                          1h2

  (xopnpk)  %open local packages%
  PROCEDURE (pknames REF, stupinfos REF, scope, pkhs REF %->
  pkhs REF%);                                                       1h2a
```

```
    %declarations%                                                    1h2a1
      LOCAL i, ub;                                                    1h2a1a
      LOCAL LIST costs [cmpkl];                                       1h2a1b
    %catchphrases%                                                    1h2a2
      (pnulst)  CATCHPHRASE; IF abrtn () THEN NULL-LISTS;             1h2a2a
      (pclspks) CATCHPHRASE; IF abr () THEN                           1h2a2b
        xclspk (&pkhs, $costs);                                       1h2a2b1
    %assure local lists released%                                     1h2a3
      INVOKE (pnulst);                                                1h2a3a
    %loop through list of package names%                              1h2a4
      ub _ pknames.L;                                                 1h2a4a
      INVOKE (pclspks);                                               1h2a4b
      FOR i _ 1 UP UNTIL > ub DO                                      1h2a4c
        #pkhs# !_                                                     1h2a4c1
          iopnpk (ELEM #pknames# [i], IF &stupinfos THEN
          ELEM #stupinfos# [i] ELSE 0, scope);                        1h2a4c1a
    %return%                                                          1h2a5
      RETURN (&pkhs);                                                 1h2a5a
      END,
                                                                      1h2a5b
  (xclspk)  %close local packages%
  PROCEDURE (pkhs REF, costs REF %-> costs REF%);                     1h2b
    %declarations%                                                    1h2b1
      LOCAL i, ub;                                                    1h2b1a
    %loop through list of package handles%                            1h2b2
      ub _ pkhs.L;                                                    1h2b2a
```

```
        FOR i _ 1 UP UNTIL > ub DO                                  1h2b2b
          #costs# !_ iclspk (ELEM #pkhs# [i]);                      1h2b2b1
      %return%                                                      1h2b3
        RETURN (&costs);                                            1h2b3a
        END,                                                        1h2b3b
%procedures%                                                        1h3
  %+PDP10% (xintpe)  %interrupt local procedure%
  PROCEDURE (ch);                                                   1h3a
    %declarations%                                                  1h3a1
      LOCAL locch;                                                  1h3a1a
      LOCAL ca POINTER;                                             1h3a1b
    %locate call record%                                            1h3a2
      ca _ fndcall (clocal, ch, cexternal : locch);                 1h3a2a
      INVOKE (palwkp,, ca);                                         1h3a2b
      IF NOT ca.capkh THEN sigerr (epfsin);                         1h3a2c
    %verify call state%                                             1h3a3
      IF ca.castate = cinterrupted THEN sigerr (epixin);            1h3a3a
    %interrupt local procedure%                                     1h3a4
      IF ca.caseqpr THEN sigerr (epfqin)                            1h3a4a
      ELSE                                                          1h3a4b
        vjusr (ca.casph, ca.capcrh, cpeint, locch);                 1h3a4b1
      ca.caprvstate _ (ca.castate := cinterrupted);                 1h3a4c
    %return%                                                        1h3a5
      RETURN;                                                       1h3a5a
```

```
        END,%+PDP10%                                                    1h3a5b

    %+PDP10% (xrsmpe)  %resume local procedure%
    PROCEDURE (ch);                                                     1h3b

      %declarations%                                                    1h3b1

        LOCAL locch;                                                    1h3b1a

        LOCAL ca POINTER;                                               1h3b1b

      %locate call record%                                              1h3b2

        ca _ fndcall (clocal, ch, cexternal : locch);                   1h3b2a

        INVOKE (palwkp,, ca);                                           1h3b2b

      %verify call state%                                               1h3b3

        IF ca.castate # cinterrupted THEN sigerr (epixrm);              1h3b3a

      %resume local procedure%                                          1h3b4

        vjusr (                                                         1h3b4a

          ca.casph, ca.capcrh, cpersm, locch);                          1h3b4a1

        ca.castate _ ca.caprvstate;                                     1h3b4b

      %return%                                                          1h3b5

        RETURN;                                                         1h3b5a

        END,%+PDP10%                                                    1h3b5b

    %+PDP10% (xabrpe)  %abort local procedure%
    PROCEDURE (ch);                                                     1h3c

      %declarations%                                                    1h3c1

        LOCAL locch, chan;                                              1h3c1a

        LOCAL ca POINTER, pr POINTER;                                   1h3c1b

      %locate call record%                                              1h3c2

        ca _ fndcall (clocal, ch, cexternal : locch);                   1h3c2a
```

```
        INVOKE (palwkp,, ca);                                   1h3c2b
        IF NOT ca.capkh THEN sigerr (epfsab);                   1h3c2c
      %abort local procedure%                                   1h3c3
        IF ca.caseqpr THEN                                      1h3c3a
          BEGIN                                                 1h3c3a1
          %locate processor record%                             1h3c3a2
            pr _ fndrec (vprfld, ca.capcrh, cshare);            1h3c3a2a
            INVOKE (palwkp,, pr);                               1h3c3a2b
          %signal processor%                                    1h3c3a3
            IF (chan _ pr.prabrchn) <= cmchno THEN              1h3c3a3a
              osigfk (pr.prfkh, chan);                          1h3c3a3a1
          END                                                   1h3c3a4
        ELSE                                                    1h3c3b
          vjusr (ca.casph, ca.capcrh, cpeabr, locch);           1h3c3b1
      %return%                                                  1h3c4
        RETURN;                                                 1h3c4a
        END,%+PDP10%
                                                                1h3c4b

(xsigpe) %signal local procedure%
PROCEDURE (ch, argl REF, arglmsk REF);                          1h3d
    %declarations%                                              1h3d1
      LOCAL acqevh, code;                                       1h3d1a
      LOCAL ca POINTER;                                         1h3d1b
    %locate call record%                                        1h3d2
      ca _ fndrec (vcafld, ch, cexternal);                      1h3d2a
      INVOKE (palwkp,, ca);                                     1h3d2b
```

```
    %verify call state%                                              1h3d3
      IF ca.castate # crunrem THEN sigerr (epixsg);                  1h3d3a
    %mask and save argument list%                                    1h3d4
      mskarlst (                                                     1h3d4a
        carglist, ca.caprml _ savent (clist, &argl),
        &arglmsk);                                                   1h3d4a1
    %notify system and local procedure%                              1h3d5
      sigecb (ca.caqecb, csiged);                                    1h3d5a
      IF acqevh _ ca.caqevh THEN                                     1h3d5b
        BEGIN                                                        1h3d5b1
        code.LH _ IF ca.caremote THEN ch ELSE ca.cacch;              1h3d5b2
          code.RH _ csiged;                                          1h3d5b2a
        sigev (acqevh, code, TRUE);                                  1h3d5b3
        END;                                                         1h3d5b4
    %wait for acknowledgment from processor%                         1h3d6
      waicok (ca.carecb);                                            1h3d6a
    %return%                                                         1h3d7
      RETURN;                                                        1h3d7a
      END,
                                                                     1h3d7b

(xntepe) %make event known to local caller%
PROCEDURE (ch, event, desc REF);                                     1h3e
    %declarations%                                                   1h3e1
      LOCAL cch, sch=0;                                              1h3e1a
      LOCAL ca POINTER, cca POINTER;                                 1h3e1b
    %catchphrases%                                                   1h3e2
```

```
    (pdelhlf) CATCHPHRASE; IF abr () AND sch THEN                    1h3e2a
       sysend (sch := 0);                                            1h3e2a1
%locate call record%                                                 1h3e3
   ca _ fndcall (cremote, ch, cexternal : cch);                      1h3e3a
   INVOKE (palwkp,, ca);                                             1h3e3b
%verify call state%                                                  1h3e4
   IF ca.castate # crunrem THEN sigerr (epixnt);                     1h3e4a
%propagate notice%                                                   1h3e5
   IF cch THEN                                                       1h3e5a
      BEGIN                                                          1h3e5a1
      %locate controlling call record%                               1h3e5a2
         cca _ fndrec (vcafld, cch, cshare);                         1h3e5a2a
         INVOKE (palwkp,, cca);                                      1h3e5a2b
      %notify controlling procedure%                                 1h3e5a3
         sch _ sysbgn (cca.caph, cxntepe, event, &desc);
                                                                     1h3e5a3a
         INVOKE (pdelhlf);                                           1h3e5a3b
      END;                                                           1h3e5a4
%notify local caller%                                                1h3e6
   vjusr (ca.casph, ca.capcrh, cpente, cch, ch, event,
   &desc);                                                           1h3e6a
%complete note propagation%                                          1h3e7
   sysend (sch := 0);                                                1h3e7a
%return%                                                             1h3e8
   RETURN;                                                           1h3e8a
   END.
                                                                     1h3e8b
```

```
(xhlppe)  %solicit help from local caller%                          1h3f
PROCEDURE (ch, pblm, desc REF %-> solution REF%);                   1h3f

   %declarations%                                                   1h3f1

      LOCAL solution, cch, provided=TRUE;                           1h3f1a

      LOCAL ca POINTER, cca POINTER;                                1h3f1b

   %catchphrases%                                                   1h3f2

      (prststate) CATCHPHRASE; IF abrtn () THEN                     1h3f2a

         ca.castate _ crunrem;                                      1h3f2a1

   %locate call record%                                             1h3f3

      ca _ fndcall (cremote, ch, cexternal : cch);                  1h3f3a

      INVOKE (palwkp,, ca);                                         1h3f3b

   %verify call state%                                              1h3f4

      IF ca.castate # crunrem THEN sigerr (epixhp);                 1h3f4a

      ca.castate _ cstuck;                                          1h3f4b

      INVOKE (prststate);                                           1h3f4c

   %solicit help from local caller%                                 1h3f5

      IF NOT ca.capkh THEN provided _ FALSE                         1h3f5a

      ELSE                                                          1h3f5b

         vjusr (ca.casph, ca.capcrh, cpehlp, cch, ch, pblm,
         &desc : solution ;pfail,, $provided);                      1h3f5b1

   %propagate request if necessary and possible%                    1h3f6

      IF NOT provided THEN                                          1h3f6a

         IF cch THEN                                                1h3f6a1

            BEGIN                                                   1h3f6a1a

            %locate controlling call record%                        1h3f6a1b

               cca _ fndrec (vcafld, cch, cshare);                  1h3f6a1b1
```

```
                INVOKE (palwkp,, cca);                              1h3f6a1b2

              %solicit help from controlling procedure%             1h3f6a1c

                syscal (cca,caph, cxhlppe, cca,cacch, pblm,
                &desc : solution);                                  1h3f6a1c1

              END                                                   1h3f6a1d

            ELSE sigerr (epfnoh);                                   1h3f6a2

        %return%                                                    1h3f7

          RETURN (solution);                                        1h3f7a

          END,                                                      1h3f7b

    %data stores%                                                   1h4

      %+PDP10% (xcrtdt)  %create local data store%
      PROCEDURE (dsel REF, value REF, scope);                       1h4a

        %declarations%                                              1h4a1

          LOCAL pk POINTER, dt POINTER;                             1h4a1a

        %locate package record%                                     1h4a2

          pk _ fndrec (vpkfld, ELEM #dsel# [2], cshare);            1h4a2a

          INVOKE (palwkp,, pk);                                     1h4a2b

        %verify package alive%                                      1h4a3

          IF pk.pkdead THEN sigerr (ekfded);                        1h4a3a

        %check for duplicate created data store name%               1h4a4

          IF fndcdt (&dsel, FALSE) THEN sigerr (eddsto);            1h4a4a

        %create new data store record%                              1h4a5

          dt _ icrtdt (&dsel, scope);                               1h4a5a

          INVOKE (puldidt,, dt);                                    1h4a5b

        %save value%                                                1h4a6
```

```
        dt,dtvalue _ savent (cblock, &value);                    1h4a6a
    %return%                                                     1h4a7
        RETURN;                                                  1h4a7a
        END,%+PDP10%                                             1h4a7b

%+PDP10% (xdeldt)  %delete local data store%
PROCEDURE (dsel REF);                                            1h4b
    %declarations%                                               1h4b1
        LOCAL dt POINTER;                                        1h4b1a
    %locate data store record%                                   1h4b2
        IF NOT dt _ fndcdt (&dsel, cdelete) THEN                 1h4b2a
            sigerr (edusto);                                     1h4b2a1
        INVOKE (puldldt,, dt);                                   1h4b2b
    %return%                                                     1h4b3
        dt,dtuse _ dt,dtuse - 2;                                 1h4b3a
        RETURN;                                                  1h4b3b
        END,%+PDP10%                                             1h4b3c

%+PDP10% (xrddt)   %read local data store%
PROCEDURE (dsel REF %-> value REF%);                             1h4c
    %declarations%                                               1h4c1
        LOCAL value, esel;                                       1h4c1a
        LOCAL pk POINTER, dt POINTER;                            1h4c1b
    %locate package record%                                      1h4c2
        pk _ fndrec (vpkfld, ELEM #dsel# [2], cshare);           1h4c2a
        INVOKE (palwkp,, pk);                                    1h4c2b
    %verify package alive%                                       1h4c3
```

```
        IF pk.pkdead THEN sigerr (ekfded);                          1h4c3a

      %lock data store if necessary%                                1h4c4

        IF dt _ fndcdt (&dsel, cshare) THEN                         1h4c4a

          BEGIN                                                     1h4c4a1

          INVOKE (puldldt,, dt);                                    1h4c4a2

          pshlcb (dt.dtlcb, cshare, TRUE);                          1h4c4a3

          INVOKE (ppoplcb);                                         1h4c4a4

          END;                                                      1h4c4a5

      %read created data store%                                     1h4c5

        esel _ ELEM #dsel# [4];                                     1h4c5a

        IF dt AND NOT dt.dtdummy THEN                               1h4c5b

          value _ savstruc (fndelm (dt.dtvalue, esel))              1h4c5b1

      %read permanent data store%                                   1h4c6

        ELSE                                                        1h4c6a

          vjusr (pk.pksph, 0, clrddt, pk.pkintpkh, ELEM
          #dsel# [3], esel : value);                                1h4c6a1

      %return%                                                      1h4c7

        IF dt THEN BUMP DOWN dt.dtuse;                              1h4c7a

        RETURN (value);                                             1h4c7b

        END.%+PDP10%
                                                                    1h4c7c

%+PDP10% (xwrdt)   %write local data store%
PROCEDURE (dsel REF, value REF);                                    1h4d

  %declarations%                                                    1h4d1

    LOCAL                                                           1h4d1a

      esel, store, ostore, ovalue, len, olen, valincr,
      ostorelen, osetpr, oset=0;                                    1h4d1a1
```

```
    LOCAL pk POINTER, dt POINTER;                                    1h4d1b
%catchphrases%                                                       1h4d2
    (prelblk) CATCHPHRASE; IF abrtn () THEN                          1h4d2a
        relent (cblock, store);                                      1h4d2a1
%locate package record%                                              1h4d3
    pk _ fndrec (vpkfld, ELEM #dsel# [2], cshare);                   1h4d3a
    INVOKE (palwkp,, pk);                                            1h4d3b
%verify package alive%                                               1h4d4
    IF pk.pkdead THEN sigerr (ekfded);                               1h4d4a
%lock data store if necessary%                                       1h4d5
    IF dt _ fndcdt (&dsel, cshare) THEN                              1h4d5a
        BEGIN                                                        1h4d5a1
        INVOKE (puldldt,, dt);                                       1h4d5a2
        pshlcb (dt.dtlcb, cexclusive, TRUE);                         1h4d5a3
        INVOKE (ppoplcb);                                            1h4d5a4
        END;                                                         1h4d5a5
%write created data store%                                           1h4d6
    esel _ ELEM #dsel# [4];                                          1h4d6a
    IF dt AND NOT dt.dtdummy THEN                                    1h4d6b
        BEGIN                                                        1h4d6b1
        %locate desired element of store%                            1h4d6b2
            ovalue _                                                 1h4d6b2a
                fndelm (ostore _ dt.dtvalue, esel : oset);           1h4d6b2a1
        %update in place if possible%                                1h4d6b3
            valincr _                                                1h4d6b3a
```

```
                ( len _ sizstruc (&value)) -                    1h4d6b3a1
                (olen _ sizstruc (ovalue));                     1h4d6b3a2
             IF valincr = 0 THEN                                 1h4d6b3b
                oblkxfr (&value, ovalue, len)                   1h4d6b3b1
          %allocate new storage block%                           1h4d6b4
             ELSE                                                1h4d6b4a
                BEGIN                                            1h4d6b4a1
                store _                                          1h4d6b4a2
                   aloent (cblock, (ostorelen _ sizent
                   (cblock, ostore)) + valincr);                1h4d6b4a2a
                INVOKE (prelblk);                                1h4d6b4a3
                oblkxfr (                                        1h4d6b4a4
                   ostore := ostore + (osetpr _ oset + olen),
                                                                 1h4d6b4a4a
                   store  := store + oset, oset);                1h4d6b4a4b
                oblkxfr (&value, store := store + len, len);
                                                                 1h4d6b4a5
                oblkxfr (ostore, store, ostorelen - osetpr);
                                                                 1h4d6b4a6
                dt,dtvalue _ (store := dt,dtvalue);              1h4d6b4a7
                END;                                             1h4d6b4a8
          END                                                    1h4d6b5
       %write permanent data store%                              1h4d7
          ELSE                                                   1h4d7a
             vjusr (pk,pksph, 0, clwrdt, pk,pkintpkh, ELEM
             #dsel# [3], esel, &value);                          1h4d7a1
       %return%                                                  1h4d8
          IF dt THEN BUMP DOWN dt,dtuse;                         1h4d8a
          RETURN;                                                1h4d8b
```

```
        END,%+PDP10%                                                    1h4d8c

    %+PDP10% (xlckdt)  %lock local data store%
    PROCEDURE (dsel REF, type, scope, wait %-> dtlh%);                   1h4e

      %declarations%                                                     1h4e1

        LOCAL ldsc, ecb;                                                 1h4e1a

        LOCAL lckid [cmlkidl];                                           1h4e1b

        LOCAL pk POINTER, dt POINTER;                                    1h4e1c

      %locate package record%                                            1h4e2

        pk _ fndrec (vpkfld, ELEM #dsel# [2], cshare);                   1h4e2a

        INVOKE (palwkp,, pk);                                            1h4e2b

      %verify package alive%                                             1h4e3

        IF pk.pkdead THEN sigerr (ekfded);                               1h4e3a

      %locate/create data store record%                                  1h4e4

        IF NOT dt _ fndcdt (&dsel, cshare) THEN                          1h4e4a

          BEGIN                                                          1h4e4a1

          %create data store record%                                     1h4e4a2

            dt _ icrtdt (&dsel, call);                                   1h4e4a2a

            dt.dtdummy _ TRUE;                                           1h4e4a2b

            INVOKE (puldldt,, dt);                                       1h4e4a2c

          %verify data store's existence%                                1h4e4a3

            vjusr (                                                      1h4e4a3a

              pk.pksph, 0, clvrdt, pk.pkintpkh, ELEM #dsel#
              [3]);                                                      1h4e4a3a1

          END                                                            1h4e4a4

        ELSE INVOKE (puldldt,, dt);                                      1h4e4b
```

```
    %allocate ECB%                                              1h4e5
       ecb _ IF wait THEN aloecb (1) ELSE 0;                    1h4e5a
       INVOKE (palwrb,, ecb);                                   1h4e5b
    %lock data store%                                           1h4e6
       lckid     _ dt.dtlcb;                                    1h4e6a
       lckid [1] _ type;                                        1h4e6b
       lckid [2] _ scope;                                       1h4e6c
       ldsc _ setlcb ($lckid, ecb, TRUE);                       1h4e6d
       waicok (ecb);                                            1h4e6e
    %return%                                                    1h4e7
       RETURN (ldsc.ldlsh);                                     1h4e7a
       END,%+PDP10%
                                                                1h4e7b
%+PDP10% (xulkdt)  %unlock local data store%
PROCEDURE (dsel REF, dtlh);                                     1h4f
    %declarations%                                              1h4f1
       LOCAL ldsc=0;                                            1h4f1a
       LOCAL pk POINTER, dt POINTER;                            1h4f1b
    %locate package record%                                     1h4f2
       pk _ fndrec (vpkfld, ELEM #dsel# [2], cshare);           1h4f2a
       INVOKE (palwkp,, pk);                                    1h4f2b
    %verify package alive%                                      1h4f3
       IF pk.pkdead THEN sigerr (ekfded);                       1h4f3a
    %locate data store record%                                  1h4f4
       IF NOT dt _ fndcdt (&dsel, cshare) THEN                  1h4f4a
          sigerr (edusto);                                      1h4f4a1
```

```
            INVOKE (puldldt,, dt);                                  1h4f4b
         %unlock data store%                                        1h4f5
            ldsc.ldlcb _ dt.dtlcb;                                  1h4f5a
            ldsc.ldlsh _ dtlh;                                      1h4f5b
            remlcb (ldsc);                                          1h4f5c
         %return%                                                   1h4f6
            dt.dtuse _ dt.dtuse - 2;                                1h4f6a
            RETURN;                                                 1h4f6b
            END.%+PDP10%                                            1h4f6c
   %channels%                                                       1h5
      (xrdmnu) %read channel-type menu%
      PROCEDURE %(-> menu REF)%;                                    1h5a
         %return%                                                   1h5a1
            RETURN (vchmnu);                                        1h5a1a
            END.                                                    1h5a1b
      (xcrtce)  %create local physical channel end%
      PROCEDURE (poh, remport REF, ph);                             1h5b
         %declarations%                                             1h5b1
            LOCAL ps POINTER;                                       1h5b1a
         %catchphrases%                                             1h5b2
            (pdelce) CATCHPHRASE; IF abr () THEN idelce (poh);      1h5b2a
         %create channel end%                                       1h5b3
            icrtce (poh, &remport);                                 1h5b3a
            INVOKE (pdelce);                                        1h5b3b
         %associate with process handle%                            1h5b4
```

```
        IF ph THEN                                                  1h5b4a
          BEGIN                                                     1h5b4a1
          %locate process record%                                   1h5b4a2
            ps _ fndrec (vpsfld, ph, cshare);                       1h5b4a2a
            INVOKE (palwkp,, ps);                                   1h5b4a2b
            IF ps.pstype # cintroduced                              1h5b4a2c
            OR ps.psadjph # vcph THEN sigerr (egmhps);              1h5b4a2d
          %check for duplicate%                                     1h5b4a3
            IF ps.pspoh THEN sigerr (esdpoh);                       1h5b4a3a
          %associate physical with logical channel%                 1h5b4a4
            ps.pspoh _ poh;                                         1h5b4a4a
          END;                                                      1h5b4a5
      %return%                                                      1h5b5
        RETURN;                                                     1h5b5a
        END.                                                        1h5b5b

(xalopo)  %allocate local physical port%
PROCEDURE (chntypmnu REF, remloc REF, active %-> poh,
mnuindex, locport REF%);                                            1h5c
    %declarations%                                                  1h5c1
      LOCAL poh, mnuindex, locport;                                 1h5c1a
    %catchphrases%                                                  1h5c2
      (prelpo) CATCHPHRASE; IF abr () THEN                          1h5c2a
        BEGIN                                                       1h5c2a1
        irelpo (poh);                                               1h5c2a2
        relent (cblock, locport);                                   1h5c2a3
```

```
        END;                                                    1h5c2a4
    %allocate port%                                             1h5c3
        poh =                                                   1h5c3a
            ialopo (&chntypmnu, &remloc, active, call :
            mnuindex, locport);                                 1h5c3a1
        INVOKE (prelpo);                                        1h5c3b
    %OK creation of channel%                                    1h5c4
        vjusr (vpsldr, 0, cokcch, poh);                         1h5c4a
    %return%                                                    1h5c5
        RETURN (poh, mnuindex, locport);                        1h5c5a
        END,                                                    1h5c5b

(xrelpo)  %[delete channel end and] release local port%
PROCEDURE (poh);                                                1h5d
    %verify port handle%                                        1h5d1
        fndrec (vpofld, poh, FALSE);                            1h5d1a
    %OK deletion of channel%                                    1h5d2
        vjusr (vpsldr, 0, cokdch, poh);                         1h5d2a
    %flush port from system%                                    1h5d3
        poflsh (poh);                                           1h5d3a
    %delete channel end (if any)%                               1h5d4
        idelce (poh);                                           1h5d4a
    %release port%                                              1h5d5
        irelpo (poh);                                           1h5d5a
    %return%                                                    1h5d6
        RETURN;                                                 1h5d6a
```

```
            END,                                                    1h5d6b

    %debugging%                                                     1h6

       (xsetrc) %set DPS trace word%
       PROCEDURE (setting);                                         1h6a

          %set trace word%                                          1h6a1

             dtrace _ setting;                                      1h6a1a

          %return%                                                  1h6a2

             RETURN;                                                1h6a2a

             END,                                                   1h6a2b

 %utilities%                                                        1i

    %manager management%                                            1i1

       (spawn) %spawn manager%
       PROCEDURE (procaddr, stupinfo, rscnt);                       1i1a

          %declarations%                                            1i1a1

             LOCAL mgh, fwbase;                                     1i1a1a

             LOCAL mg POINTER;                                      1i1a1b

          %advance meter%                                           1i1a2

             BUMP vmmgrc;                                           1i1a2a

          %create manager record%                                   1i1a3

             mg _ crtrec (vmgfld, call : mgh);                      1i1a3a

             INVOKE (prtnkp,, mg);                                  1i1a3b

             mg.mgroot   _ mg.mgaddr _ procaddr;                    1i1a3c

             mg.mgstup   _ stupinfo;                                1i1a3d

             mg.mgrscnt  _ rscnt;                                   1i1a3e

             mg.mgfwbase _ fwbase _ cl110r*(mgh-cmgmnh);            1i1a3f
```

```
        %spawning self%                                          111a4
           IF NOT procaddr THEN                                  111a4a
              BEGIN                                              111a4a1
              vmg _ mg;                                          111a4a2
              vmgh _ mgh;                                        111a4a3
              vacs _ vmg,mgacs;                                  111a4a4
              mgcover ();                                        111a4a5
              END;                                               111a4a6
        %return%                                                 111a5
           RETURN;                                               111a5a
           END.                                                  111a5b

     (mgcover) %manager initiator%
     PROCEDURE;                                                  111b
        %declarations%                                           111b1
           LOCAL addr, i;                                        111b1a
           LOCAL nxtpage REF;                                    111b1b
        %intercept terminal aborts%                              111b2
           INVOKE (pcover);                                      111b2a
        %reacquaint L10 with storage zone%                       111b3
           lstzone _ vstgzon;                                    111b3a
        %force L10 runtime state private%                        111b4
           FOR i _ 0 UP UNTIL > cl110r-1 DO                      111b4a
              BEGIN                                              111b4a1
              &nxtpage _ cpglen*(cpl10r + i) + cpglen-1;         111b4a2
                 %touching AC 0 buys nothing%                    111b4a2a
```

```
            nxtpage _ nxtpage;                                          111b4a3
            END;                                                        111b4a4
        %save L10 state%                                                111b5
          ocnnfk (0, cpl10r, vl10jfn, vmg.mgfwbase, cll10r);            111b5a
        %return if spawning self%                                       111b6
          IF NOT addr _ (vmg.mgaddr := 0) THEN RETURN;                  111b6a
        %loop through allowed restarts of manager%                      111b7
          FOR vmg.mgurscnt _ 1 UP UNTIL > vmg.mgrscnt DO                111b7a
            [addr] (vmg.mgstup);                                        111b7a1
        %verify lock stack empty%                                       111b8
          IF vlckstk THEN sigerr (elistk);                              111b8a
        %return%                                                        111b9
          owait (0);                                                    111b9a
          END,                                                          111b9b
    (selmgr) %select next manager for dispatch%
    PROCEDURE;                                                          111c
      %declarations%                                                    111c1
        LOCAL                                                           111c1a
          port, mgcount, i, j, nxtlen, ub, index, ph, pcrh,
          fkh;                                                          111c1a1
        LOCAL nxtecb REF, nxtecbs REF;                                  111c1b
      %run as wheel%                                                    111c2
        INVOKE (prstwhl,, vwheel := TRUE);                              111c2a
      %select next manager for consideration%                           111c3
        OPENPORT runfld (vmgfld, FALSE : [port]);                       111c3a
```

```
        WHILE                                                   1i1c3b
           (vnewmg _ PCALL [port] (clock))                      1i1c3b1
           AND vnewmg # vnxtmg DO NULL;                         1i1c3b2
        IF NOT vnewmg THEN PCALL [port] (crewind);              1i1c3c
     %select next manager for dispatch%                         1i1c4
        vnxtmg _ vindex _ 0;                                    1i1c4a
        LOOP                                                    1i1c4b
           BEGIN                                                1i1c4b1
           %search for dispatchable manager%                    1i1c4b2
              mgcount _ sizfld (vmgfld);                        1i1c4b2a
              FOR j _ 1 UP UNTIL > mgcount DO                   1i1c4b2b
                 BEGIN                                          1i1c4b2b1
                 %locate next manager's record%                 1i1c4b2b2
                    WHILE NOT                                   1i1c4b2b2a
                       vnewmg _ PCALL [port] (clock : vnewh)
                                                                1i1c4b2b2a1
                    DO PCALL [port] (crewind);                  1i1c4b2b2b
                    IF NOT vnxtmg THEN vnxtmg _ vnewmg;         1i1c4b2b2c
                 %select manager if new%                        1i1c4b2b3
                    IF vnewmg.mgaddr THEN EXIT LOOP 2;          1i1c4b2b3a
                 %select manager if event signalled%            1i1c4b2b4
                    IF NOT &nxtecbs _ vnewmg.mgecbs THEN        1i1c4b2b4a
                       REPEAT LOOP;                             1i1c4b2b4a1
                    vlength _ 0;                                1i1c4b2b4b
                    %ECB list%                                  1i1c4b2b4c
                       IF vnewmg.mglon THEN                     1i1c4b2b4c1
```

```
                BEGIN                                   111c4b2b4c1a
                ub _ nxtecbs.L;                         111c4b2b4c1b
                FOR i _ 1 UP UNTIL > ub DO              111c4b2b4c1c
                   BEGIN                                111c4b2b4c1c1
                   &nxtecb _ ELEM #nxtecbs# [i];
                                                        111c4b2b4c1c2
                   otest (&nxtecb, FALSE : nxtlen);
                                                        111c4b2b4c1c3
                   vlength _ vlength + nxtlen;
                                                        111c4b2b4c1c4
                   IF NOT vindex AND nxtlen             111c4b2b4c1c5
                   AND vindex _ 1 THEN                  111c4b2b4c1c6
                      vcode _ otest (&nxtecb, TRUE);
                                                       111c4b2b4c1c6a
                   END;                                 111c4b2b4c1c7
                END                                     111c4b2b4c1d
             %single ECB%                                111c4b2b4d
                ELSE                                    111c4b2b4d1
                   IF vcode _ otest (&nxtecbs, TRUE :
                   vlength) THEN vindex _ 1;            111c4b2b4d1a
             IF vindex THEN EXIT LOOP 2;                111c4b2b4e
          END;                                          111c4b2b5
       %wait DPS%                                       111c4b3
          owtsig ();                                    111c4b3a
       %assimilate external events%                     111c4b4
          LOOP CASE otest ($vwtdecb, TRUE) OF           111c4b4a
             = 0: EXIT LOOP;                            111c4b4a1
             = 1: %processor activity%                  111c4b4a2
                WHILE fkh _ ofndfk () DO                111c4b4a2a
```

```
                    IF pcrh _ fndpr (fkh) THEN          1ii1c4b4a2a1
                       spawn ($oprmgr, pcrh, 1)         1ii1c4b4a2a1a
                    ELSE orsmfk (fkh, -1);              1ii1c4b4a2a2
                 = 2: %port activity%                   1ii1c4b4a3
                    BEGIN                               1ii1c4b4a3a
                    %update state of affected port(s)%  1ii1c4b4a3b
                       iupdpo (0);                      1ii1c4b4a3b1
                    %service remote processes (if any)% 1ii1c4b4a3c
                       WHILE ph _ fndrps () DO          1ii1c4b4a3c1
                          spawn ($msgmgr, ph, 1);       1ii1c4b4a3c1a
                    END;                                1ii1c4b4a3d
                 = 3: %fork termination%                1ii1c4b4a4
                    dedprs ();                          1ii1c4b4a4a
                 = 4: %interval timer expiration%       1ii1c4b4a5
                    updtmr ();                          1ii1c4b4a5a
                 ENDCASE sigerr (esuwtd);               1ii1c4b4a6
              END;                                      1ii1c4b5
           vnewmg.mgecbs _ 0;                           1ii1c4c
           vmgroot _ vnewmg.mgroot;                     1ii1c4d
        %return%                                        1ii1c5
           RETURN;                                      1ii1c5a
           END.
                                                        1ii1c5b
%process management%                                    1i2
   (syscal) %call remote system procedure%
```

```
PROCEDURE (ph, number, arg1, arg2, arg3, arg4, arg5, arg6,
arg7 %-> EMPTY, res1, res2, res3%);                          1i2a

   %declarations%                                            1i2a1

      LOCAL rawres1 [cmrperes1];                             1i2a1a

      LOCAL LIST                                             1i2a1b

         psel [cmpsel1], arg1 [cmrpeargl], res1 [cmrperes1];
                                                             1i2a1b1
      LOCAL entry POINTER;                                   1i2a1c

   %catchphrases%                                            1i2a2

      (pnulst) CATCHPHRASE; IF abrtn () THEN NULL-LISTS;     1i2a2a

   %assure local lists released%                             1i2a3

      INVOKE (pnulst);                                       1i2a3a

   %construct procedure selector%                            1i2a4

      #psel# _ ph, 0, number;                                1i2a4a

   %construct argument list%                                 1i2a5

      #arg1# _ arg1, arg2, arg3, arg4, arg5, arg6, arg7;     1i2a5a

      entry _ fndsyp (number);                               1i2a5b

      #arg1# _ MOVE #arg1# [1 TO entry.enargc];              1i2a5c

   %call remote system procedure%                            1i2a6

      IF calpe ($psel, csyprio, $arg1, 0, 0, $res1, entry) #
      csuccess THEN                                          1i2a6a

         sigerr (epiotc);                                    1i2a6a1

   %decompose result list%                                   1i2a7

      mkarray ($res1, $rawres1, TRUE);                       1i2a7a

   %return%                                                  1i2a8

      RETURN (0, rawres1, rawres1 [1], rawres1 [2]);         1i2a8a
```

```
    END,                                                          1i2a8b

(sysbgn) %begin remote system procedure%
PROCEDURE (ph, number, arg1, arg2, arg3, arg4, arg5, arg6,
arg7 %-> ch%);                                                    1i2b

  %declarations%                                                  1i2b1

    LOCAL ch;                                                     1i2b1a

    LOCAL LIST psel [cmpsell], argl [cmrpeargl];                  1i2b1b

    LOCAL entry POINTER;                                          1i2b1c

  %catchphrases%                                                  1i2b2

    (pnulst) CATCHPHRASE; IF abrtn () THEN NULL-LISTS;            1i2b2a

    (prelch) CATCHPHRASE; IF abr   () THEN relch (ch);            1i2b2b

  %assure local lists released%                                   1i2b3

    INVOKE (pnulst);                                              1i2b3a

  %construct procedure selector%                                  1i2b4

    #psel# _ ph, 0, number;                                       1i2b4a

  %allocate call handle%                                          1i2b5

    ch _ aloch (spsel, csyprio, entry _ fndsyp (number));         1i2b5a

    INVOKE (prelch);                                              1i2b5b

  %construct argument list%                                       1i2b6

    #argl# _ arg1, arg2, arg3, arg4, arg5, arg6, arg7;            1i2b6a

    #argl# _ MOVE #argl# [1 TO entry,enargc];                     1i2b6b

  %initiate procedure%                                            1i2b7

    relpe (ch, $argl, 0, 0, 0);                                   1i2b7a

  %return%                                                        1i2b8

    RETURN (ch);                                                  1i2b8a
```

```
        END,                                                    1i2b8b

    (sysend) %resume remote system procedure%
    PROCEDURE (ch %-> EMPTY, res1, res2, res3%);                1i2c

      %declarations%                                            1i2c1

        LOCAL rawres1 [cmrperes1];                              1i2c1a

        LOCAL LIST res1 [cmrperes1];                            1i2c1b

      %catchphrases%                                            1i2c2

        (pnulst) CATCHPHRASE; IF abrtn () THEN NULL-LISTS;      1i2c2a

      %assure local lists released%                             1i2c3

        INVOKE (pnulst);                                        1i2c3a

      %acquire control from system procedure%                   1i2c4

        IF acqpe (ch, $res1) # csuccess THEN                    1i2c4a

          sigerr (epiotc);                                      1i2c4a1

      %release call handle%                                     1i2c5

        relch (ch);                                             1i2c5a

      %decompose result list%                                   1i2c6

        mkarray ($res1, $rawres1, TRUE);                        1i2c6a

      %return%                                                  1i2c7

        RETURN (0, rawres1, rawres1 [1], rawres1 [2]);          1i2c7a

        END,                                                    1i2c7b

    (sndps) %send message to process%
    PROCEDURE (ph, msg REF);                                    1i2d

      %declarations%                                            1i2d1

        LOCAL poh, block;                                       1i2d1a

        LOCAL ctx [cmctx1];                                     1i2d1b
```

```
    LOCAL ps POINTER;                                    112d1c
%advance meter%                                          112d2
    BUMP vmsndc;                                         112d2a
%locate process record%                                  112d3
    ps _ fndrec (vpsfld, ph, csend);                     112d3a
    INVOKE (palwkp,, ps);                                112d3b
    poh _ ps.pspoh;                                      112d3c
%select routing address%                                 112d4
    IF NOT ELEM #msg# [cmrout] THEN                      112d4a
        #msg# [cmrout] _                                 112d4a1
            IF poh THEN cfph ELSE ps.psadjsh;            112d4a1a
%encode message%                                         112d5
    block _ encstruc (cmessage, &msg);                   112d5a
    INVOKE (palwrb,, block);                             112d5b
%self%                                                   112d6
    IF ph = vfph THEN                                    112d6a
        BEGIN                                            112d6a1
        %trace message%                                  112d6a2
            IF dtrace .A ctrfmsg THEN                    112d6a2a
                trcpo (0, 0, block);                     112d6a2a1
        %save context%                                   112d6a3
            oblkxfr ($vctx, $ctx, cmctxl);               112d6a3a
            INVOKE (prstctx,, $ctx);                     112d6a3b
        %send and receive message%                       112d6a4
            vfmsg _ block;                               112d6a4a
```

```
            msgmgr (vfph);                                  1i2d6a4b
         END                                                1i2d6a5
      %non-self%                                            1i2d7
         ELSE                                               1i2d7a
            BEGIN                                           1i2d7a1
            %locate adjacent process%                       1i2d7a2
               IF NOT poh THEN                              1i2d7a2a
                  BEGIN                                     1i2d7a2a1
                  ps _ fndrec (vpsfld, ps.psadjph, csend);  1i2d7a2a2
                  INVOKE (palwkp,, ps);                     1i2d7a2a3
                  poh _ ps.pspoh;                           1i2d7a2a4
                  END;                                      1i2d7a2a5
            %verify process alive%                          1i2d7a3
               IF ps.psdead THEN sigerr (esfded);           1i2d7a3a
            %send the message%                              1i2d7a4
               isndch (poh, block);                         1i2d7a4a
            END;                                            1i2d7a5
      %return%                                              1i2d8
         RETURN;                                            1i2d8a
         END,
                                                            1i2d8b
(rcvps) %receive message from adjacent process%
PROCEDURE (ph, msg REF %-> msg REF%);                       1i2e
      %declarations%                                        1i2e1
         LOCAL poh, block;                                  1i2e1a
         LOCAL ps POINTER;                                  1i2e1b
```

```
        %advance meter%                                          112e2
           BUMP vmrcvc;                                          112e2a
        %locate process record%                                  112e3
           ps _ fndrec (vpsfld, ph, creceive);                   112e3a
           INVOKE (palwkp,, ps);                                 112e3b
        %self%                                                   112e4
           IF ph = vfph THEN block _ (vfmsg := 0)                112e4a
        %non-self%                                               112e5
           ELSE                                                  112e5a
              BEGIN                                              112e5a1
              %verify adjacent%                                  112e5a2
                 IF NOT poh _ ps.pspoh THEN sigerr (esircv);     112e5a2a
              %verify process alive%                             112e5a3
                 IF ps.psdead THEN sigerr (esfded);              112e5a3a
              %accept message%                                   112e5a4
                 block _ ircvch (poh);                           112e5a4a
                 INVOKE (palwrb,, block);                        112e5a4b
              END;                                               112e5a5
        %decode message%                                         112e6
           decstruc (cmessage, block, &msg);                     112e6a
        %return%                                                 112e7
           RETURN (&msg);                                        112e7a
           END,                                                  112e7b

     %processor management%                                      113
```

```
(vjusr) %call user%
PROCEDURE (sph, pcrh, number %optional priority in LH%,
arg1, arg2, arg3, arg4, arg5, arg6, arg7 %-> cost, res1,
res2, res3%);                                                    1i3a

   %signals%                                                     1i3a1

      % NOTE (cselpr, pr, pcrh) %                                1i3a1a

   %declarations%                                                1i3a2

      LOCAL                                                      1i3a2a

         ush, selpcrh, priority, port, priook, nxtpcrh,
         wtecbs, code, i, ecb, cost;                             1i3a2a1

      LOCAL rawres1 [cmuscres1];                                 1i3a2b

      LOCAL arg1 REF, res1 REF;                                  1i3a2c

      LOCAL entry POINTER, us POINTER, su POINTER, pr
      POINTER;                                                   1i3a2d

   %catchphrases%                                                1i3a3

      (punlcksu) CATCHPHRASE; IF abrtn () THEN                   1i3a3a

         BEGIN                                                   1i3a3a1

         rececb (wtecbs, ecb);                                   1i3a3a2

         unlckrec (su, ckeep);                                   1i3a3a3

         END;                                                    1i3a3a4

      (prelpr)   CATCHPHRASE; IF abr () THEN                     1i3a3b

         BEGIN                                                   1i3a3b1

         BUMP pr.prrycnt;                                        1i3a3b2

         WHILE code _ otest (wtecbs, TRUE) DO                    1i3a3b3

            sigecb (code, cok);                                  1i3a3b3a

         END;                                                    1i3a3b4

   %advance meter%                                               1i3a4
```

```
    BUMP vmvjuc;                                                    1i3a4a

%NOP if no process leader%                                          1i3a5

    IF NOT vpsldr THEN RETURN (0);                                  1i3a5a

%lookup user call%                                                  1i3a6

    IF NOT (number,RH IN [1, cmuscn]) THEN sigerr
    (eruusc);                                                       1i3a6a

    entry _ $dusctbl + 5*(number,RH-1);                             1i3a6b

%select priority%                                                   1i3a7

    priority _ IF number,LH THEN number,LH ELSE
    entry,enpadr;                                                   1i3a7a

%construct argument list%                                           1i3a8

    &arg1 _ aloent (clist, cmuscarg1);                              1i3a8a

    INVOKE (palwrl,, &arg1);                                        1i3a8b

    &res1 _ aloent (clist, cmuscres1);                              1i3a8c

    INVOKE (palwrl,, &res1);                                        1i3a8d

    #arg1# _ arg1, arg2, arg3, arg4, arg5, arg6, arg7;              1i3a8e

    #arg1# _ MOVE #arg1# [1 TO entry,enargc];                       1i3a8f

%create user call record%                                           1i3a9

    us _ crtrec (vusfld, call : ush);                               1i3a9a

    INVOKE (palwdl,, us);                                           1i3a9b

    us,usnumber _ number,RH;                                        1i3a9c

    us,usstate  _ cuncalled;                                        1i3a9d

    us,uscph    _ vcph;                                             1i3a9e

    us,ussph    _ sph;                                              1i3a9f

    us,usentry  _ entry;                                            1i3a9g

    us,usarg1   _ &arg1;                                            1i3a9h
```

```
    us,usresl   _ &resl;                                    1i3a9i
    ecb _ us,usecb;                                         1i3a9j
  %locate subprocess record%                                1i3a10
    su _ fndrec (vsufld, sph, cshare);                      1i3a10a
    wtecbs _ su,suwtecbs;                                   1i3a10b
    INVOKE(punlcksu);                                       1i3a10c
    IF NOT su,susignin THEN sigerr (erisin);                1i3a10d
  %specific processor%                                      1i3a11
    IF selpcrh _ pcrh THEN                                  1i3a11a
      BEGIN                                                 1i3a11a1
      %locate processor record%                             1i3a11a2
        pr _ fndrec (vprfld, pcrh, cshare);                 1i3a11a2a
        INVOKE (palwkp,, pr);                               1i3a11a2b
        IF pr,prdead THEN sigerr (erfded);                  1i3a11a2c
      %Verify processor's suitability%                      1i3a11a3
        IF NOT pr,prsignin THEN sigerr (erisin);            1i3a11a3a
        IF priority < pr,prprio THEN sigerr (epfpio);       1i3a11a3b
      %wait for processor to ready ifself%                  1i3a11a4
        IF priority # cimmprio THEN                         1i3a11a4a
          WHILE NOT pr,prrycnt DO                           1i3a11a4a1
            BEGIN                                           1i3a11a4a1a
            sigecb (wtecbs, ecb);                           1i3a11a4a1b
            waicok (ecb);                                   1i3a11a4a1c
            END;                                            1i3a11a4a1d
      END                                                   1i3a11a5
```

```
%any processor%                                                    113a12
   ELSE                                                            113a12a
      BEGIN                                                        113a12a1
      OPENPORT runfld (vprfld, cshare : [port]);                   113a12a2
      LOOP                                                         113a12a3
         BEGIN                                                     113a12a3a
         %declare need for processor%                              113a12a3b
            sigecb (wtecbs, ecb);                                  113a12a3b1
         %search for suitable processor%                           113a12a3c
            priook _ FALSE;                                        113a12a3c1
            WHILE pr _ PCALL [port] (cprevue : nxtpcrh)
            DO                                                     113a12a3c2
               IF pr.prsph = sph                                   113a12a3c2a
               AND NOT pr.prsecond                                 113a12a3c2b
               AND NOT pr.prdead                                   113a12a3c2c
               AND priority >= pr.prprio                           113a12a3c2d
               AND (priook _ TRUE)                                 113a12a3c2e
               AND PCALL [port] (clock)                            113a12a3c2f
               AND ((priority = cimmprio OR pr.prrycnt)
                                                                   113a12a3c2g
               AND selpcrh _ nxtpcrh) THEN                         113a12a3c2h
                  EXIT LOOP 2;                                     113a12a3c2h1
         %create new processor if necessary%                       113a12a3d
            IF su.suautopcr AND su.sucrnpr < su.sumxnpr
            THEN                                                   113a12a3d1
               BEGIN                                               113a12a3d1a
               crtpr (sph, 0, call, priority, FALSE);
                                                                   113a12a3d1b
```

```
                    rececb (wtecbs, ecb);                    113a12a3d1c
                    END                                      113a12a3d1d
                ELSE                                         113a12a3d2
                    IF priook THEN waicok (ecb)              113a12a3d2a
                    ELSE sigerr (epfpio);                    113a12a3d2b
            %rewind folder%                                  113a12a3e
                PCALL [port] (crewind);                      113a12a3e1
            END;                                             113a12a3f
        rececb (wtecbs, ecb);                                113a12a4
        END;                                                 113a12a5
%allocate processor%                                         113a13
    IF priority # cimmprio THEN                              113a13a
        BEGIN                                                113a13a1
        BUMP DOWN pr,prrycnt;                                113a13a2
        INVOKE (prelpr);                                     113a13a3
        END;                                                 113a13a4
    us,uspcrh _ selpcrh;                                     113a13b
%notify caller of processor selected%                        113a14
    NOTE (cselpr, pr, selpcrh);                              113a14a
%solicit service from processor%                             113a15
    chgrec (us, cshare);                                     113a15a
    IF pr,prseqpr THEN sigecb (pr,prsiecb, ush)              113a15b
    ELSE                                                     113a15c
        BEGIN                                                113a15c1
        code,LH _ ush; code,RH _ number,RH;                  113a15c2
```

```
            sigev (pr.prjuevh, code, TRUE);                    1i3a15c3
            END;                                               1i3a15c4
         IF priority # cimmprio THEN DROP (prelpr);            1i3a15d
      %wait for completion of user call%                       1i3a16
         code _ waiecb (ecb);                                  1i3a16a
         IF code # cok THEN                                    1i3a16b
            BEGIN                                              1i3a16b1
            *verrmsg* _ *[us,usdgmsg]*;                        1i3a16b2
            ABORT (code, $verrmsg);                            1i3a16b3
            END;                                               1i3a16b4
      %decompose result list%                                  1i3a17
         mkarray (&resl, $rawresl, TRUE);                      1i3a17a
      %return%                                                 1i3a18
         RETURN (us.uscost, rawresl, rawresl [1], rawresl [2]);
                                                               1i3a18a
         END,
                                                               1i3a18b

(encpr) %encode processor parameters%
PROCEDURE (pr POINTER, acs REF, prml REF, cnt, specs REF);     1i3b
   %declarations%                                              1i3b1
      LOCAL bp, i, spec, parm;                                 1i3b1a
      LOCAL dst REF;                                           1i3b1b
   %verify number of parameters%                               1i3b2
      IF prml.L # cnt THEN sigerr (edwpmc);                    1i3b2a
   %initialize resource block%                                 1i3b3
      IF cnt THEN inirb (pr);                                  1i3b3a
   %loop through parameters%                                   1i3b4
```

```
        bp _ opntwrd (&specs, cprpmspcl);                          113b4a
        FOR i _ 1 UP UNTIL > cnt DO                                113b4b
          BEGIN                                                    113b4b1
          %fetch parameter spec%                                   113b4b2
            spec _ ordbyt ($bp);                                   113b4b2a
          %output parameter to processor%                          113b4b3
            parm _ wrpr (spec.spdtype, pr, ELEM #prm1# [i]);       113b4b3a
          %deposit parameter token in processor's AC%              113b4b4
            &dst _ &acs + spec.spac + 1;                           113b4b4a
            CASE spec.sploc OF                                     113b4b4b
              = crh: dst.RH _ parm;                                113b4b4b1
              = clh: dst.LH _ parm;                                113b4b4b2
              = cwh: dst    _ parm;                                113b4b4b3
              = cfg: dst.LH _ parm.LH;                             113b4b4b4
              ENDCASE sigerr (erupm1);                             113b4b4b5
          END;                                                     113b4b5
      %terminate resource block%                                   113b5
        IF cnt THEN trmrb (pr);                                    113b5a
      %return%                                                     113b6
        RETURN;                                                    113b6a
        END.
                                                                   113b6b

(decpr) %decode processor parameters%
PROCEDURE (pr POINTER, acs REF, prm1 REF, cnt, specs REF);         113c
      %declarations%                                               113c1
        LOCAL bp, i, spec, parm, type;                             113c1a
```

```
        LOCAL src REF;                                          1i3c1b
      %initialize parameter list%                               1i3c2
        #prm1# _;                                               1i3c2a
      %loop through parameters%                                 1i3c3
        bp _ opntwrd (&specs, cprpmspcl);                       1i3c3a
        FOR i _ 1 UP UNTIL > cnt DO                             1i3c3b
          BEGIN                                                 1i3c3b1
          %fetch parameter spec%                                1i3c3b2
            spec _ ordbyt ($bp);                                1i3c3b2a
          %extract parameter token from processor's AC%         1i3c3b3
            &src _ &acs + spec.spac + 1;                        1i3c3b3a
            CASE spec.sploc OF                                  1i3c3b3b
              = crh: parm _ src.RH;                             1i3c3b3b1
              = clh: parm _ src.LH;                             1i3c3b3b2
              = cwh, = cfg: parm _ src;                         1i3c3b3b3
              ENDCASE sigerr (erupm1);                          1i3c3b3b4
          %input parameter from processor%                      1i3c3b4
            type _ spec.spdtype;                                1i3c3b4a
            #prm1# !_ USE rdpr (type, pr, parm, -1);            1i3c3b4b
          END;                                                  1i3c3b5
      %return%                                                  1i3c4
        RETURN;                                                 1i3c4a
        END,
                                                                1i3c4b
    %+PDP10% (rdpr) %read data from processor%
```

```
PROCEDURE (type, pr POINTER, src, occasionaldst %-> dst
REF%);                                                          113d

   %declarations%                                               113d1

      LOCAL                                                     113d1a

         funtype, claimedlen, len, intsrc, i,
         subtype=cstruc, descr=(occasionaldst = -1);            113d1a1

      LOCAL dst REF =0;                                         113d1b

   %catchphrases%                                               113d2

      (prelelm) CATCHPHRASE; IF abr () THEN freelm (&dst);      113d2a

   %determine fundamental type%                                 113d3

      funtype _ type,tyfunt;                                    113d3a

      IF type,tyempok AND NOT src THEN funtype _ cempty         113d3b

      ELSE IF type,tylist THEN subtype _ (funtype := clist);    113d3c

   %dispatch via fundamental type%                              113d4

      INVOKE (prelelm);                                         113d4a

      CASE funtype OF                                           113d4b

         = cempty, = cdumemp:                                   113d4b1

            &dst _ IF descr                                     113d4b1a

               THEN makelm (lnull)                              113d4b1a1

               ELSE 0;                                          113d4b1a2

         = cindex:                                              113d4b2

            IF src IN [1, cmxidx] THEN REPEAT CASE
            (cinteger)                                          113d4b2a

            ELSE sigerr (ediidx);                               113d4b2b

         = cinteger:                                            113d4b3

            &dst _ IF descr                                     113d4b3a
```

```
        THEN makelm (linteg, src)                                   1i3d4b3a1
        ELSE src;                                                   1i3d4b3a2
    = ccharstr, = cucstr:                                           1i3d4b4
      BEGIN                                                         1i3d4b4a
      %locate source%                                               1i3d4b4b
        intsrc _ rdpras (ccharstr, pr, src : len);                  1i3d4b4b1
      %allocate destination%                                        1i3d4b4c
        &dst _ aloent (ccharstr, len);                              1i3d4b4c1
      %copy character string%                                       1i3d4b4d
        ostrxfr (intsrc, opntstr (&dst), dst,L _
        len);                                                       1i3d4b4d1
      %force upper case%                                            1i3d4b4e
        IF funtype = cucstr THEN                                    1i3d4b4e1
          *dst* _ + SF(*dst*) SE(*dst*);                            1i3d4b4e1a
      %create descriptor%                                           1i3d4b4f
        IF descr THEN &dst _ descrb (&dst);                         1i3d4b4f1
      END;                                                          1i3d4b4g
    = cstruc, =cdata:                                               1i3d4b5
      BEGIN                                                         1i3d4b5a
      %accept structure from processor%                             1i3d4b5b
        IF src THEN                                                 1i3d4b5b1
          BEGIN                                                     1i3d4b5b1a
          %locate source%                                           1i3d4b5b1b
            intsrc _                                                1i3d4b5b1b1
              rdpras (cblock, pr, src :
              claimedlen);                                          1i3d4b5b1b1a
```

```
                %verify data structure%                          1i3d4b5b1c
                  len _ IF funtype = cstruc                      1i3d4b5b1c1
                    THEN sizstruc (intsrc) ELSE
                    claimedlen;                                  1i3d4b5b1c1a
                  IF len > claimedlen THEN sigerr
                  (edoabc);                                      1i3d4b5b1c2
                %allocate destination%                           1i3d4b5b1d
                  &dst _ aloent (cblock, len);                   1i3d4b5b1d1
                %copy block%                                     1i3d4b5b1e
                  oblkxfr (intsrc, &dst, len);                   1i3d4b5b1e1
                END                                              1i3d4b5b1f
              %generate EMPTY for processor%                      1i3d4b5c
                ELSE &dst _ encstruc (cempty);                   1i3d4b5c1
              %create descriptor%                                1i3d4b5d
                IF descr THEN &dst _ descrb (&dst);              1i3d4b5d1
              END;                                               1i3d4b5e
            = clist;                                             1i3d4b6
              BEGIN                                              1i3d4b6a
              %locate source%                                    1i3d4b6b
                intsrc _ rdpras (cblock, pr, src : len);         1i3d4b6b1
              %allocate destination%                             1i3d4b6c
                &dst _ aloent (clist, len);                      1i3d4b6c1
              %copy list element tokens%                         1i3d4b6d
                oblkxfr (intsrc, &dst+1, len);                   1i3d4b6d1
              %check length%                                     1i3d4b6e
                IF len > cmlistl THEN sigerr (edolst);           1i3d4b6e1
```

```
        IF funtype = cusrinf AND len # cmusrinfl THEN          1i3d4b6e2
           sigerr (ediuif);                                    1i3d4b6e2a

     %read list elements%                                      1i3d4b6f

        FOR i _ 1 UP UNTIL > len DO                            1i3d4b6f1

           #dst# [i] _ USE rdpr (subtype, pr, dst
           [i], -i);                                           1i3d4b6f1a

     %create descriptor%                                       1i3d4b6g

        IF descr THEN &dst _ descrl (&dst);                    1i3d4b6g1

     END;                                                      1i3d4b6h

  = cusrinf:                                                   1i3d4b7

     BEGIN                                                     1i3d4b7a

     subtype _ cucstr;                                         1i3d4b7b

     REPEAT CASE (clist);                                      1i3d4b7c

     END;                                                      1i3d4b7d

  = cintpsel, = cdsel;                                         1i3d4b8

     BEGIN                                                     1i3d4b8a

     %locate source%                                           1i3d4b8b

        intsrc _ rdpras (cblock, pr, src : len);               1i3d4b8b1

     %allocate destination%                                    1i3d4b8c

        &dst _ aloent (clist, len);                            1i3d4b8c1

     %copy list element tokens%                                1i3d4b8d

        oblkxfr (intsrc, &dst+1, len);                         1i3d4b8d1

     %check length%                                            1i3d4b8e

        IF len # (IF type = cintpsel THEN cmpsell
        ELSE cmdsell) THEN                                     1i3d4b8e1

           sigerr (edipdl);                                    1i3d4b8e1a
```

```
        %read list elements%                                    1i3d4b8f
          #dst# _                                               1i3d4b8f1
            USE rdpr (cindex,   pr, dst [1], -1),               1i3d4b8f1a
            USE rdpr (cindex,   pr, dst [2], -1),               1i3d4b8f1b
            USE rdpr (cucstr, pr, dst [3], -1);                 1i3d4b8f1c
          IF type = cdsel THEN                                  1i3d4b8f2
            #dst# !_ USE rdpr (cstruc, pr, dst [4],
            -1);                                                1i3d4b8f2a
        %construct descriptor%                                  1i3d4b8g
          IF descr THEN &dst _ descr1 (&dst);                   1i3d4b8g1
        END;                                                    1i3d4b8h
      = csmlblk:                                                1i3d4b9
        BEGIN                                                   1i3d4b9a
        %locate source%                                         1i3d4b9b
          intsrc _ rdpras (cblock, pr, src : len);              1i3d4b9b1
        %check length%                                          1i3d4b9c
          IF len > cmsmlblk THEN sigerr (erosm1);               1i3d4b9c1
        %copy small block%                                      1i3d4b9d
          oblkxfr (intsrc, &dst _ occasionaldst, len);
                                                                1i3d4b9d1
        END;                                                    1i3d4b9e
      = cdumlist:                                               1i3d4b10
        BEGIN                                                   1i3d4b10a
        %allocate destination%                                  1i3d4b10b
          &dst _ aloent (clist, cmdum11);                       1i3d4b10b1
        %construct descriptor%                                  1i3d4b10c
```

```
                IF descr THEN &dst _ descrl (&dst);          1i3d4b10c1
              END;                                           1i3d4b10d
            ENDCASE sigerr (erurde);                         1i3d4b11
        %return%                                             1i3d5
          RETURN (&dst);                                     1i3d5a
          END,%+PDP10%
                                                             1i3d5b
%+PDP10% (wrpr) %write data to processor%
PROCEDURE (type, pr POINTER, src REF %-> dst%);              1i3e
        %declarations%                                       1i3e1
          LOCAL funtype, dst, intdst, len, i, subtype=cstruc;  1i3e1a
          LOCAL rawlist [cmlist1];                           1i3e1b
        %determine fundamental type%                         1i3e2
          funtype _ type.tyfunt;                             1i3e2a
          IF type.tyempok AND NOT &src THEN funtype _ cempty 1i3e2b
          ELSE IF type.tylist THEN subtype _ (funtype := clist); 1i3e2c
        %dispatch via fundamental type%                      1i3e3
          CASE funtype OF                                    1i3e3a
            = cempty:                                        1i3e3a1
              dst _ 0;                                       1i3e3a1a
            = cindex:                                        1i3e3a2
              IF &src IN [1, cmxidx] THEN REPEAT CASE
              (cinteger)                                     1i3e3a2a
              ELSE sigerr (ediidx);                          1i3e3a2b
            = cinteger:                                      1i3e3a3
              dst _ &src;                                    1i3e3a3a
```

```
= ccharstr:                                                    1i3e3a4
   BEGIN                                                       1i3e3a4a
   intdst _ wrpras (ccharstr, pr, len _ src.L :
   dst);                                                       1i3e3a4b
   ostrxfr (opntstr (&src), intdst, len);                      1i3e3a4c
   END;                                                        1i3e3a4d
= cstruc, = cdata, = calostruc, = calodata:                    1i3e3a5
   BEGIN                                                       1i3e3a5a
   intdst _                                                    1i3e3a5b
      wrpras (cblock, pr, len _ sizent (cblock,
      &src) : dst);                                            1i3e3a5b1
   oblkxfr (&src, intdst, len);                                1i3e3a5c
   END;                                                        1i3e3a5d
= clist:                                                       1i3e3a6
   BEGIN                                                       1i3e3a6a
   IF (len _ src.L) > cmlist1 THEN sigerr (edolst);
                                                               1i3e3a6b
   FOR i _ 1 UP UNTIL > len DO                                 1i3e3a6c
      rawlist [i-1] _                                          1i3e3a6c1
         wrpr (subtype, pr, ELEM #src# [i]);                   1i3e3a6c1a
   intdst _ wrpras (cblock, pr, len : dst);                    1i3e3a6d
   oblkxfr (srawlist, intdst, len);                            1i3e3a6e
   END;                                                        1i3e3a6f
= cusrinf:                                                     1i3e3a7
   BEGIN                                                       1i3e3a7a
   IF src.L # cmusrinf1 THEN sigerr (ediuif);                  1i3e3a7b
   subtype _ ccharstr;                                         1i3e3a7c
```

```
                REPEAT CASE (clist);                                    1i3e3a7d
                END;                                                    1i3e3a7e
             = cintpsel, = cdsel:                                       1i3e3a8
                BEGIN                                                   1i3e3a8a
                IF src.L # (len _ IF type = cintpsel THEN
                cmpsell ELSE cmdsell) THEN                              1i3e3a8b
                   sigerr (edipdl);                                     1i3e3a8b1
                rawlist     _ wrpr (cindex,   pr, ELEM #src#
                [1]);                                                   1i3e3a8c
                rawlist [1] _ wrpr (cindex,   pr, ELEM #src#
                [2]);                                                   1i3e3a8d
                rawlist [2] _ wrpr (ccharstr, pr, ELEM #src#
                [3]);                                                   1i3e3a8e
                IF type = cdsel THEN                                    1i3e3a8f
                   rawlist [3] _                                        1i3e3a8f1
                      wrpr (cstruc, pr, ELEM #src# [4]);                1i3e3a8f1a
                intdst _ wrpras (cblock, pr, len : dst);                1i3e3a8g
                oblkxfr (srawlist, intdst, len);                        1i3e3a8h
                END;                                                    1i3e3a8i
             ENDCASE sigerr (eruwre);                                   1i3e3a9
          %return%                                                      1i3e4
             RETURN (dst);                                              1i3e4a
             END,%+PDP10%
                                                                        1i3e4b
%+PDP10% (rdpras) %view entity in processor's address space%
PROCEDURE (type, pr POINTER, praddr %-> cfaddr REF,
length%);                                                               1i3f
   %declarations%                                                       1i3f1
```

```
    LOCAL                                                          1i3f1a

       length, prpg, cfpg, cfaddr, offset, fkh, algnlen,
       pgincr, bp, more;                                           1i3f1a1

%verify address%                                                   1i3f2

   IF NOT praddr,RH THEN sigerr (emiadr);                          1i3f2a

%map in first page of entity%                                      1i3f3

   DIV praddr,RH / cpglen, prpg, offset;                           1i3f3a

   cfaddr _ cpprwn*cpglen + offset;                                1i3f3b

   ocnnfk (fkh _ pr.prfkh, prpg, 0, cfpg _ cpprwn, 1);             1i3f3c

%dispatch via entity type%                                         1i3f4

   CASE type OF                                                    1i3f4a

      = cblock:                                                    1i3f4a1

         BEGIN                                                     1i3f4a1a

         %map in additional pages if required%                     1i3f4a1b

            IF (algnlen _ offset + 1 + (length _ [cfaddr
            := cfaddr+1],L)) > clprwn*cpglen THEN                  1i3f4a1b1

               sigerr (erowin);                                    1i3f4a1b1a

            IF pgincr _ (algnlen+cpglen-1)/cpglen - 1
            THEN                                                   1i3f4a1b2

               ocnnfk (fkh, prpg+1, 0, cfpg+1, pgincr);
                                                                   1i3f4a1b2a
         END;                                                      1i3f4a1c

      = ccharstr:                                                  1i3f4a2

         BEGIN                                                     1i3f4a2a

         %create return byte pointer%                              1i3f4a2b

            cfaddr         _ praddr;                               1i3f4a2b1

            cfaddr.adpage _ cpprwn;                                1i3f4a2b2
```

```
            IF cfaddr.LH = 777777B THEN cfaddr.LH _
            440700B;                                        1i3f4a2b3

         %verify byte pointer%                              1i3f4a2c

            IF cfaddr.bpindx                                1i3f4a2c1

            OR cfaddr.bpsize # CHAR THEN                    1i3f4a2c2

               sigerr (emibyp);                             1i3f4a2c2a

         %map in character string%                          1i3f4a2d

            bp _ cfaddr;                                    1i3f4a2d1

            length _ 0;                                     1i3f4a2d2

            DO length _ length +                            1i3f4a2d3

               ostrsiz ($bp, cpglen*(cfpg _ cfpg+1) :
               more)                                        1i3f4a2d3a

            WHILE                                           1i3f4a2d4

               more AND ocnnfk (fkh, prpg _ prpg+1, 0,
               cfpg, 1);                                    1i3f4a2d4a

         END;                                               1i3f4a2e

      ENDCASE sigerr (erualo);                              1i3f4a3

   %return%                                                 1i3f5

      RETURN (cfaddr, length);                              1i3f5a

      END,%+PDP10%
                                                            1i3f5b

%+PDP10% (wrpras) %allocate entity in processor's address
space%
PROCEDURE (type, pr POINTER, length %-> cfaddr, praddr%);   1i3g

   %declarations%                                           1i3g1

      LOCAL                                                 1i3g1a

         intlength, curlen, prpg, offset, praddr, cfaddr,
         pgincr;                                            1i3g1a1
```

```
%compute size of entity to be allocated%                    1i3g2

   intlength _                                              1i3g2a

      IF type = ccharstr THEN (length+5)/5 ELSE length+1;
                                                           1i3g2a1
%resource block overflow%                                   1i3g3

   IF (pr.prrbl _ (curlen _ pr.prrbl) + intlength) >
   pr.prrbm AND pr.prrbovflw _ TRUE THEN                    1i3g3a

      RETURN (0, 0);                                        1i3g3a1

%map in allocated space%                                    1i3g4

   DIV                                                      1i3g4a

      (praddr _ pr.prrbaddr+1+curlen) / cpglen, prpg,
      offset;                                               1i3g4a1

   cfaddr _ cpprwn*cpglen + offset;                         1i3g4b

   IF (pgincr _ (offset+intlength+cpglen-1)/cpglen) >
   clprwn THEN                                              1i3g4c

      sigerr (erowin);                                      1i3g4c1

   ocnnfk (pr.prfkh, prpg, 0, cpprwn, pgincr);              1i3g4d

%dispatch via entity type%                                  1i3g5

   CASE type OF                                             1i3g5a

      = cblock: [cfaddr := cfaddr+1] _ length*1B6 +
      length;                                               1i3g5a1

      = ccharstr:                                           1i3g5a2

         BEGIN                                              1i3g5a2a

         cfaddr _ MKFD (WORD, 7, cfaddr);                   1i3g5a2b

         praddr _ MKFD (WORD, 7, praddr);                   1i3g5a2c

         END;                                               1i3g5a2d

      ENDCASE sigerr (erualo);                              1i3g5a3

%return%                                                    1i3g6
```

```
        RETURN (cfaddr, praddr);                                    1i3g6a

        END,%+PDP10%                                                1i3g6b

%+PDP10% (inirb) %initialize resource block%                        1i3h
PROCEDURE (pr POINTER);

   %declarations%                                                   1i3h1

      LOCAL rb REF;                                                 1i3h1a

   %initialize resource block%                                      1i3h2

      &rb _ rdpras (cblock, pr, pr.prrbaddr) - 1;                   1i3h2a

      pr.prrbm      _ rb.M;                                         1i3h2b

      pr.prrbl      _ 0;                                            1i3h2c

      pr.prrbovflw _ FALSE;                                         1i3h2d

   %return%                                                         1i3h3

      RETURN;                                                       1i3h3a

      END,%+PDP10%                                                  1i3h3b

%+PDP10% (trmrb) %terminate resource block%                         1i3i
PROCEDURE (pr POINTER);

   %declarations%                                                   1i3i1

      LOCAL rb REF;                                                 1i3i1a

   %terminate resource block%                                       1i3i2

      &rb _ rdpras (cblock, pr, pr.prrbaddr) - 1;                   1i3i2a

      rb.L _ pr.prrbl;                                              1i3i2b

      IF pr.prrbovflw THEN sigerr (erorsb);                         1i3i2c

   %return%                                                         1i3i3

      RETURN;                                                       1i3i3a
```

```
        END,%+PDP10%                                                    11313b

    %+PDP10% (rsmpr) %resume processor after DPS operation%
    PROCEDURE (pr POINTER, outcome, acs REF);                           113j

      %declarations%                                                    113j1

        LOCAL fkh;                                                      113j1a

      %NOP if already running%                                          113j2

        IF NOT pr.prrunning THEN                                        113j2a

          BEGIN                                                         113j2a1

          %update ACs%                                                  113j2a2

            owracs (fkh = pr.prfkh, &acs);                              113j2a2a

          %discard resource block%                                      113j2a3

            pr.prrbaddr = 0;                                            113j2a3a

          %resume fork%                                                 113j2a4

            orsmfk (fkh, IF outcome = csuccess THEN 1 ELSE
            0);                                                         113j2a4a

            pr.prrunning = TRUE;                                        113j2a4b

          END;                                                          113j2a5

      %return%                                                          113j3

        RETURN;                                                         113j3a

        END,%+PDP10%                                                    113j3b

    %+PDP10% (dedprs) %dispose of newly dead processors%
    PROCEDURE;                                                          113k

      %declarations%                                                    113k1

        LOCAL port, fkh;                                                113k1a

        LOCAL pr POINTER;                                               113k1b
```

```
        %locate dead processors%                                    1i3k2
            OPENPORT runfld (vprfld, FALSE : [port]);               1i3k2a
            WHILE pr _ PCALL [port] (clock) DO                      1i3k2b
               IF (fkh _ pr.prfkh)                                  1i3k2b1
               AND NOT ostsfk (fkh)                                 1i3k2b2
               AND NOT (pr.prdead := TRUE) THEN NULL;               1i3k2b3
        %return%                                                    1i3k3
            RETURN;                                                 1i3k3a
            END.%+PDP10%
                                                                    1i3k3b
    %inter-process communication%                                   1i4
        (icrtps) %create process%
        PROCEDURE (psaddr REF, userinfo REF, stupinfo REF, scope,
        intrahostaddr REF %-> poh, intrahostaddr REF%);             1i4a
            %declarations%                                          1i4a1
               LOCAL poh, intaction;                                1i4a1a
               LOCAL STRING action [cmactl], host [cmhsnl];         1i4a1b
               LOCAL LIST remloc [3];                               1i4a1c
               LOCAL TEXT POINTER tp1, tp2, tp3, tp4, tp5, tp6;     1i4a1d
               LOCAL po pOINTER;                                    1i4a1e
            %catchphrases%                                          1i4a2
               (pnulst) CATCHPHRASE; IF abrtn () THEN NULL-LISTS;   1i4a2a
               (pdelps) CATCHPHRASE; IF abr   () THEN               1i4a2b
                  [selipc (po, cidelps)] (po);                      1i4a2b1
            %assure local lists released%                           1i4a3
               INVOKE (pnulst);                                     1i4a3a
```

```
%parse process address%                                              114a4
   IF NOT                                                            114a4a
      FIND SF(*psaddr*) ^tp1 1$PT ^tp2 1$SP ^tp3 1$PT
      ^tp4 (1$SP ^tp5 1$PT / ^tp5) ^tp6 ENDCHR THEN                  114a4a1
         sigerr (esipsa);                                            114a4a1a
   IF tp5 [1] = tp6 [1] THEN                                         114a4b
      BEGIN                                                          114a4b1
      tp3 [1] _ (tp5 [1] := tp3 [1]);                                114a4b2
      tp4 [1] _ (tp6 [1] := tp4 [1]);                                114a4b3
      END;                                                           114a4b4
   *action*         _ tp1 tp2;                                       114a4c
   *host*           _ tp3 tp4;                                       114a4d
   *intrahostaddr*  _ tp5 tp6;                                       114a4e
%decode action%                                                      114a5
   intaction _                                                       114a5a
      IF *action* = "CRT" THEN ccreate ELSE                          114a5a1
      IF *action* = "SPL" THEN csplice ELSE                          114a5a2
         sigerr (esipsa);                                            114a5a2a
%create port record%                                                 114a6
   po _ crtrec (vpofld, scope : pon);                                114a6a
   INVOKE (prtnkp,, po);                                             114a6b
   po.poactive _ TRUE;                                               114a6c
   po.poformat _ cb36;                                               114a6d
   po.potype   _ IF host.L OR intaction = csplice                    114a6e
      THEN cintrhst                                                  114a6e1
```

```
            ELSE cintrfrk;                                          114a6e2

        %create process%                                            114a7

            [selipc (po, cicrtps)]                                  114a7a

                (po, intaction, $host, &intrahostaddr, &userinfo,
                &stupinfo, $remloc);                                114a7a1

            INVOKE (pdelps);                                        114a7b

        %return%                                                    114a8

            po.pocreated _ TRUE;                                    114a8a

            RETURN (poh);                                           114a8b

            END,
                                                                    114a8c

    (idelps) %delete process%
    PROCEDURE (poh);                                                114b

        %declarations%                                              114b1

            LOCAL po POINTER;                                       114b1a

        %locate port record%                                        114b2

            po _ fndrec (vpofld, poh, cdelete);                     114b2a

            INVOKE (prtndl,, po);                                   114b2b

        %delete process%                                            114b3

            [selipc (po, cidelps)] (po);                            114b3a

        %return%                                                    114b4

            RETURN;                                                 114b4a

            END,
                                                                    114b4b

    (icrtce) %create local channel end%
    PROCEDURE (poh, remport REF);                                   114c

        %declarations%                                              114c1
```

```
        LOCAL po POINTER;                                          1i4c1a
    %locate port record%                                           1i4c2
        po _ fndrec (vpofld, poh, cexclusive);                     1i4c2a
        INVOKE (palwkp,, po);                                      1i4c2b
    %check for duplicate%                                          1i4c3
        IF po.pocreated THEN sigerr (ecdcce);                      1i4c3a
    %create end of inter-process channel%                          1i4c4
        [selipc (po, cicrtce)] (po, &remport);                     1i4c4a
    %return%                                                       1i4c5
        po.pocreated _ TRUE;                                       1i4c5a
        RETURN;                                                    1i4c5b
        END,
                                                                   1i4c5c

(idelce) %delete local channel end%
PROCEDURE (poh);                                                   1i4d
    %declarations%                                                 1i4d1
        LOCAL po POINTER;                                          1i4d1a
    %locate port record%                                           1i4d2
        po _ fndrec (vpofld, poh, cexclusive);                     1i4d2a
        INVOKE (palwkp,, po);                                      1i4d2b
    %delete end of inter-process channel%                          1i4d3
        IF po.pocreated THEN [selipc (Po, cidelce)] (po);          1i4d3a
    %return%                                                       1i4d4
        po.pocreated _ FALSE;                                      1i4d4a
        RETURN;                                                    1i4d4b
```

```
        END,                                                    1i4d4c

    (ialopo) %allocate local port%
    PROCEDURE (menu REF, remloc REF, active, scope %-> poh,
    mnuindex, locport REF%);                                    1i4e

      %declarations%                                            1i4e1

        LOCAL poh, i, item, type, format, ub, locport, j=0;     1i4e1a

        LOCAL po POINTER;                                       1i4e1b

      %create port record%                                      1i4e2

        po _ crtrec (vpofld, scope : poh);                      1i4e2a

        INVOKE (prtnkp,, po);                                   1i4e2b

        po.poactive _ active;                                   1i4e2c

      %select channel type from menu%                           1i4e3

        ub _ menu.L;                                            1i4e3a

        FOR i _ 1 UP UNTIL > ub DO                              1i4e3b

          CASE type _ decstruc (cindex, item _ ELEM #menu#
          [i], 0) OF                                            1i4e3b1

            = cintrfrk, = cintrjob: NULL;                       1i4e3b1a

            = cintrhst:                                         1i4e3b1b

              IF (j _ i)                                        1i4e3b1b1

              AND (format _ deckey (cindex, item)) = cb36
              THEN                                              1i4e3b1b2

                EXIT LOOP;                                      1i4e3b1b2a

            ENDCASE sigerr (ecutyp);                            1i4e3b1c

        IF NOT j THEN sigerr (ecwmnu);                          1i4e3c

      %record selection%                                        1i4e4

        po.potype   _ cintrhst;                                 1i4e4a
```

```
        po.poformat _ format;                                   1i4e4b
    %allocate port%                                             1i4e5
        locport _ [selipc (po, cialopo)] (po, &remloc);         1i4e5a
    %return%                                                    1i4e6
        RETURN (poh, j, locport);                               1i4e6a
        END,
                                                                1i4e6b

(irelpo) %release local port%
PROCEDURE (poh);                                                1i4f
    %declarations%                                              1i4f1
        LOCAL po POINTER;                                       1i4f1a
    %locate port record%                                        1i4f2
        po _ fndrec (vpofld, poh, cdelete);                     1i4f2a
        INVOKE (prtnd1,, po);                                   1i4f2b
    %release port%                                              1i4f3
        [selipc (po, cirelpo)] (po);                            1i4f3a
    %return%                                                    1i4f4
        RETURN;                                                 1i4f4a
        END,
                                                                1i4f4b

(isndch) %send data on channel%
PROCEDURE (poh, block REF);                                     1i4g
    %declarations%                                              1i4g1
        LOCAL po POINTER;                                       1i4g1a
    %locate port record%                                        1i4g2
        po _ fndrec (vpofld, poh, csend);                       1i4g2a
        INVOKE (palwkp,, po);                                   1i4g2b
```

```
    %trace message%                                                 1i4g3
       IF dtrace .A ctromsg THEN                                    1i4g3a
          trcpo (csend, poh, &block);                               1i4g3a1
    %send message%                                                  1i4g4
       [selipc (po, cisndch)] (po, &block);                         1i4g4a
    %return%                                                        1i4g5
       RETURN;                                                      1i4g5a
       END,                                                         1i4g5b

(ircvch) %receive data from channel%
PROCEDURE (poh %-> block REF%);                                     1i4h
    %declarations%                                                  1i4h1
       LOCAL block;                                                 1i4h1a
       LOCAL po POINTER;                                            1i4h1b
    %locate port record%                                            1i4h2
       po _ fndrec (vpofld, poh, creceive);                         1i4h2a
       INVOKE (palwkp,, po);                                        1i4h2b
    %receive message%                                               1i4h3
       block _ [selipc (po, circvch)] (po);                         1i4h3a
    %trace message%                                                 1i4h4
       IF dtrace .A ctrimsg THEN                                    1i4h4a
          trcpo (creceive, poh, block);                             1i4h4a1
    %return%                                                        1i4h5
       RETURN (block);                                              1i4h5a
       END,                                                         1i4h5b
```

```
(iupdpo) %update port state%
PROCEDURE (poh %-> aliveornot%);                                    1i4i

   %declarations%                                                   1i4i1

      LOCAL port, aliveornot=TRUE;                                  1i4i1a

      LOCAL po POINTER;                                             1i4i1b

   %update one port's state%                                        1i4i2

      IF poh THEN                                                   1i4i2a

         BEGIN                                                      1i4i2a1

         po _ fndrec (vpofld, poh, FALSE);                          1i4i2a2

         aliveornot _                                               1i4i2a3

            [selipc (po, ciupdpo)] (po);                            1i4i2a3a

         END                                                        1i4i2a4

   %update all ports' states%                                       1i4i3

      ELSE                                                          1i4i3a

         BEGIN                                                      1i4i3a1

         vwhlpls _ TRUE;                                            1i4i3a2

         OPENPORT runfld (vpofld, FALSE : [port]);                  1i4i3a3

         WHILE po _ PCALL [port] (clock) DO                         1i4i3a4

            [selipc (po, ciupdpo)] (po);                            1i4i3a4a

         END;                                                       1i4i3a5

   %return%                                                         1i4i4

      RETURN (aliveornot);                                          1i4i4a

      END,                                                          1i4i4b

(itstpo) %test local port for incoming message%
PROCEDURE (poh %-> outcome%);                                       1i4j
```

```
        %declarations%                                                  1i4j1
           LOCAL po POINTER;                                            1i4j1a
        %locate port record%                                            1i4j2
           po _ fndrec (vpofld, poh, FALSE);                            1i4j2a
        %return%                                                        1i4j3
           RETURN (po.pocreated AND otest (po.porcvecb, TRUE));         1i4j3a
           END,
                                                                        1i4j3b
     (selipc) %select IPC subroutine%
     PROCEDURE (po POINTER, number %-> procaddr%);                      1i4k
        %return%                                                        1i4k1
           RETURN (dipctbl [cmipcn*(po.potype-1) + number-1]);          1i4k1a
           END,
                                                                        1i4k1b
  %locators%                                                            1i5
     (fndcall) %locate procedure call record%
     PROCEDURE (calltype, ch, type %-> ca, otherch%);                   1i5a
        %declarations%                                                  1i5a1
           LOCAL port, otherch, cch, ph;                                1i5a1a
           LOCAL ca POINTER;                                            1i5a1b
        %search call folder for call record%                            1i5a2
           OPENPORT runfld (vcafld, type : [port]);                     1i5a2a
           WHILE ca _ PCALL [port] (cprevue : otherch) DO               1i5a2b
              BEGIN                                                     1i5a2b1
              cch _ ca.cacch;                                           1i5a2b2
              IF ((calltype = clocal AND cch = ch)                      1i5a2b3
```

```
                OR (calltype = cremote AND (otherch := cch) =
                ch))                                              1i5a2b3a

             AND ((ph _ ca.caph) = vcph                           1i5a2b4

                OR rdrec (cph, ca) = vcph OR vwheel)              1i5a2b4a

             AND PCALL [port] (cuse) THEN                         1i5a2b5

                RETURN (ca, otherch);                             1i5a2b5a

             END;                                                 1i5a2b6

          %return%                                                1i5a3

             sigerr (egmhca);                                     1i5a3a

             END,
                                                                  1i5a3b

       %+PDP10% (fndcdt) %locate data store record%
       PROCEDURE (dsel REF, type %-> dt POINTER%);                1i5b

          %declarations%                                          1i5b1

             LOCAL hashcode, port, dname, outcome=FALSE;          1i5b1a

             LOCAL dt POINTER;                                    1i5b1b

          %hash data store name%                                  1i5b2

             hashcode _ hash (dname _ ELEM #dsel# [3]);           1i5b2a

          %loop through package descriptors%                      1i5b3

             OPENPORT runfld (vdtfld, type : [port]);             1i5b3a

             WHILE dt _ PCALL [port] (cprevue) DO                 1i5b3b

                IF dt.dthash = hashcode AND *[dt.dtname]* = *dname*
                AND PCALL [port] (cuse) THEN                      1i5b3b1

                   BEGIN                                          1i5b3b1a

                   IF type THEN BUMP dt.dtuse;                    1i5b3b1b

                   outcome _ dt;                                  1i5b3b1c

                   EXIT LOOP;                                     1i5b3b1d
```

```
            END;                                                    115b3b1e
    %return%                                                        115b4
        RETURN (outcome);                                           115b4a
        END,%+PDP10%
                                                                    115b4b
(fndsyp) %locate system procedure table entry%
PROCEDURE (number %-> entry POINTER%);                              115c
    %verify procedure number%                                       115c1
        IF NOT (number IN [1, cmsypn]) THEN sigerr (epusyn);        115c1a
    %return%                                                        115c2
        RETURN ($dsyptbl + 5*(number-1));                           115c2a
        END,
                                                                    115c2b
(fndmsg) %locate message table entry%
PROCEDURE (number %-> entry POINTER%);                              115d
    %verify message number%                                         115d1
        IF NOT (number IN [1, cmmsgn]) THEN sigerr (esumsg);        115d1a
    %return%                                                        115d2
        RETURN ($dmsgtbl + 6*(number-1));                           115d2a
        END,
                                                                    115d2b
(fnddsd) %locate informal data structure definition%
PROCEDURE (type %-> entry POINTER%);                                115e
    %verify data type%                                              115e1
        IF NOT (type IN [cmnlcdt, cmxlcdt]) THEN                    115e1a
            sigerr (eduity);                                        115e1a1
    %return%                                                        115e2
        RETURN ($ddsdtbl + 4*(type-cmnlcdt));                       115e2a
```

```
        END,                                                    115e2b

    %+PDP10% (fndpr) %locate processor with specified fork
    handle%
    PROCEDURE (fkh %-> pcrh%);                                  115f

       %declarations%                                           115f1

          LOCAL port, pcrh, outcome=FALSE;                      115f1a

          LOCAL pr POINTER;                                     115f1b

       %locate process with specified fork handle%              115f2

          OPENPORT runfld (vprfld, FALSE : [port]);             115f2a

          WHILE pr _ PCALL [port] (clock : pcrh) DO             115f2b

             IF pr.prfkh = fkh AND outcome _ pcrh THEN EXIT
             LOOP;                                              115f2b1

       %return%                                                 115f3

          RETURN (outcome);                                     115f3a

          END,%+PDP10%
                                                                115f3b

    (fndrps) %locate process with incoming message%
    PROCEDURE %(-> ph)%;                                        115g

       %declarations%                                           115g1

          LOCAL port, ph, poh, outcome=FALSE;                   115g1a

          LOCAL ps POINTER;                                     115g1b

       %locate process with message pending%                    115g2

          OPENPORT runfld (vpsfld, FALSE : [port]);             115g2a

          WHILE ps _ PCALL [port] (clock : ph) DO               115g2b

             IF (poh _ ps.pspoh)                                115g2b1

             AND itstpo (poh) AND outcome _ ph THEN             115g2b2

                EXIT LOOP;                                      115g2b2a
```

```
    %return%                                                    1i5g3
      RETURN (outcome);                                         1i5g3a
      END,                                                      1i5g3b

(fnderm) %locate error message for DPS error%
PROCEDURE (number %-> msg REF%);                                1i5h
    %declarations%                                              1i5h1
      LOCAL class, subcode, msg;                                1i5h1a
      LOCAL base REF;                                           1i5h1b
    %verify error number%                                       1i5h2
      IF NOT (number IN [1, emxern]) THEN                       1i5h2a
        sigerr (eeuern);                                        1i5h2a1
    %select error message%                                      1i5h3
      DIV number / 50, class, subcode;                          1i5h3a
      BUMP class;                                               1i5h3b
      IF dermsgs                                                1i5h3c
      AND class IN [1, dertbl]                                  1i5h3d
      AND (&base _ dertbl [class])                              1i5h3e
      AND subcode IN [1, base] THEN                             1i5h3f
        msg _ base [subcode]                                    1i5h3f1
      ELSE                                                      1i5h3g
        BEGIN                                                   1i5h3g1
        *verrmsg* _ "DPS error number ", STRING (number),
        ".;                                                     1i5h3g2
        msg _ $verrmsg;                                         1i5h3g3
        END;                                                    1i5h3g4
```

```
        %return%                                                    1i5h4
           RETURN (msg);                                            1i5h4a
           END,
                                                                    1i5h4b
     %folder and record management%                                 1i6
        (crtfld) %create folder%
        PROCEDURE (def POINTER %-> fld POINTER%);                   1i6a
           %declarations%                                           1i6a1
              LOCAL map, mapl;                                      1i6a1a
              LOCAL fld POINTER;                                    1i6a1b
           %allocate folder%                                        1i6a2
              fld _                                                 1i6a2a
                 aloent (cblock, fldr.SIZE + (mapl _ def.dfmapl));  1i6a2a1
              fld.fddef _ def;                                      1i6a2b
           %initialize handle bit map%                              1i6a3
              [map _ fld + fldr.SIZE] _ -1;                         1i6a3a
              IF mapl > 1 THEN                                      1i6a3b
                 oblkxfr (map, map+1, mapl-1);                      1i6a3b1
           %return%                                                 1i6a4
              RETURN (fld);                                         1i6a4a
              END,
                                                                    1i6a4b
        (delfld) %delete folder%
        PROCEDURE (fld POINTER);                                    1i6b
           %declarations%                                           1i6b1
              LOCAL port;                                           1i6b1a
              LOCAL rec POINTER;                                    1i6b1b
```

```
        %disable creation of new records%                          116b2
           fld.fddrain _ TRUE;                                     116b2a
        %delete all records%                                       116b3
           vwhlpls _ TRUE;                                         116b3a
           OPENPORT runfld (fld, cdelete : [port]);                116b3b
           WHILE rec _ PCALL [port] (clock) DO delrec (rec);       116b3c
        %release folder%                                           116b4
           relent (cblock, fld);                                   116b4a
        %return%                                                   116b5
           RETURN;                                                 116b5a
           END,                                                    116b5b

     (runfld) %run folder%
     COROUTINE (fld POINTER, type);                                116c
        %declarations%                                             116c1
           LOCAL                                                   116c1a
              rech, wheel, oldop, prevued=FALSE, locked=FALSE,
              op=0;                                                116c1a1
           LOCAL                                                   116c1b
              rec POINTER, nxtintrec POINTER, intrec POINTER
              =fld;                                                116c1b1
        %catchphrases%                                             116c2
           (pcleanup) CATCHPHRASE; IF abrtn () THEN                116c2a
              BEGIN                                                116c2a1
              %unlock record%                                      116c2a2
                 IF locked AND op # cuse THEN unlckrec (rec,
                 ckeep);                                           116c2a2a
```

```
        %decrement use count; expunge deleted records%          1i6c2a3
          IF NOT fld.fduse _ fld.fduse-1 THEN                   1i6c2a3a
            BEGIN                                               1i6c2a3a1
            intrec _ nxtintrec _ fld.fdchnf;                    1i6c2a3a2
            WHILE intrec _ (nxtintrec :=
            nxtintrec.rcchnf) DO                                1i6c2a3a3
              IF intrec.rcdeleted THEN                          1i6c2a3a3a
                delrec (intrec + crcofst);                      1i6c2a3a3a1
            END;                                                1i6c2a3a4
        END;                                                    1i6c2a4
    %port entry%                                                1i6c3
      PORT ENTRY                                                1i6c3a
        BEGIN                                                   1i6c3a1
        %advance meter%                                         1i6c3a2
          BUMP vm.nfc;                                          1i6c3a2a
        %bump folder's use count%                               1i6c3a3
          BUMP fld.fduse;                                       1i6c3a3a
          INVOKE (pcleanup);                                    1i6c3a3b
        %enable if caller so requests%                          1i6c3a4
          IF vwhlpls := FALSE THEN                              1i6c3a4a
            BEGIN                                               1i6c3a4a1
            wheel _ (vwheel := TRUE);                           1i6c3a4a2
            INVOKE (prstwhl,, wheel);                           1i6c3a4a3
            END;                                                1i6c3a4a4
        END EXIT op _ PCALL;                                    1i6c3a5
```

```
%perform requested operation%                                    1i6c4
   LOOP CASE op OF                                               1i6c4a
      = crewind:                                                 1i6c4a1
         BEGIN                                                   1i6c4a1a
         intrec _ fld;                                           1i6c4a1b
         prevued _ locked _ FALSE;                               1i6c4a1c
         op _ PCALL;                                             1i6c4a1d
         END;                                                    1i6c4a1e
      = cprevue, = clock, = cuse:                                1i6c4a2
         BEGIN                                                   1i6c4a2a
         %check for overrun%                                     1i6c4a2b
            IF NOT intrec THEN sigerr (eforun);                  1i6c4a2b1
         %locate next accessible, non-deleted record%            1i6c4a2c
            IF NOT (prevued := TRUE) OR op = cprevue THEN
                                                                 1i6c4a2c1
               BEGIN                                             1i6c4a2c1a
               DO intrec _ intrec.rcchnf UNTIL                   1i6c4a2c1b
                  NOT intrec                                     1i6c4a2c1b1
                  OR                                             1i6c4a2c1b2
                     (NOT intrec.rcdeleted                       1i6c4a2c1b2a
                     AND                                         1i6c4a2c1b2b
                        (vwheel                                  1i6c4a2c1b2b1
                        OR winscope (                            1i6c4a2c1b2b2
                           $vcpcrid, [intrec,rclckid+2],
                           intrec.rcpcrid)));                    1i6c4a2c1b2b2a
               IF intrec THEN                                    1i6c4a2c1c
```

```
                    BEGIN                                    1i6c4a2c1c1
                    rec _ intrec + crcofst;                  1i6c4a2c1c2
                    rech _ intrec.rchnd;                     1i6c4a2c1c3
                    END                                      1i6c4a2c1c4
                 ELSE rec _ 0;                               1i6c4a2c1d
                 END;                                        1i6c4a2c1e
              %lock record%                                  1i6c4a2d
                 IF (op # cprevue AND prevued := FALSE)      1i6c4a2d1
                 AND rec AND (type AND locked _ TRUE) THEN   1i6c4a2d2
                    lckrec (rec, type ;pfail,, $rec);        1i6c4a2d2a
              %deliver record to caller%                     1i6c4a2e
                 oldop _ (op := PCALL ((rec).RH, rech));     1i6c4a2e1
              %unlock record%                                1i6c4a2f
                 IF locked := FALSE AND oldop # cuse THEN    1i6c4a2f1
                    unlckrec (rec, ckeep);                   1i6c4a2f1a
              END;                                           1i6c4a2g
           ENDCASE sigerr (efurop);                          1i6c4a3
     %return%                                                1i6c5
        END,
                                                             1i6c5a
(sizfld) %compute size of folder%
PROCEDURE (fld POINTER %-> reccount%);                       1i6d
     %return%                                                1i6d1
        RETURN (fld.fdreccnt);                               1i6d1a
        END,
                                                             1i6d1b
```

```
(crtrec) %create record%
PROCEDURE (fld POINTER, scope %-> rec POINTER, rech%);                    1i6e

  %declarations%                                                          1i6e1

    LOCAL map, mapl, rech, intrech, pcrid, addr;                          1i6e1a

    LOCAL lckid REF;                                                      1i6e1b

    LOCAL                                                                 1i6e1c

      rec POINTER, intrec POINTER, brec POINTER, def
      POINTER;                                                            1i6e1c1

  %catchphrases%                                                          1i6e2

    (prelhnd)   CATCHPHRASE; IF abr () THEN                               1i6e2a

      orelbit (map, mapl, intrech);                                       1i6e2a1

    (ptrmrec)   CATCHPHRASE; IF abr () THEN [addr] (rec);                 1i6e2b

    (punlckrec) CATCHPHRASE; IF abr () THEN                               1i6e2c

      unlckrec (rec, ckeep);                                              1i6e2c1

  %advance meter%                                                         1i6e3

    BUMP vmcrfc;                                                          1i6e3a

  %allocate record%                                                       1i6e4

    def _ fld.fddef;                                                      1i6e4a

    intrec _                                                              1i6e4b

      aloent (cblock, crcofst + [def.dfrcdf].RH +
      def.dfrcwl);                                                        1i6e4b1

    INVOKE (pabrrb,, intrec);                                             1i6e4c

    intrec.rcfld _ fld;                                                   1i6e4d

    oblkxfr (                                                             1i6e4e

      $vcpcrid,                                                           1i6e4e1

      intrec.rcpcrid _ pcrid _ intrec + recr.SIZE,
      cmpidl);                                                            1i6e4e2
```

```
    intrec,rclckid _ &lckid _ pcrid + cmpidl;                    116e4f
    inilcb (intrec,rclcb _ lckid _ &lckid + cmlkidl);            116e4g
    lckid [2] _ scope;                                           116e4h
    rec _ intrec + crcofst;                                      116e4i
%assign handle%                                                  116e5
    IF NOT (intrech _                                            116e5a
       oalobit (map _ fld+fldr,SIZE, mapl _ def,dfmapl))         116e5a1
    OR (intrec,rchnd _ rech _ intrech+def,dfmnhd-1) >
    def,dfmxhd THEN                                              116e5b
       sigerr (def,dfenovf);                                     116e5b1
    INVOKE (prelhnd);                                            116e5c
%perform user initialization%                                    116e6
    IF addr _ def,dfinit THEN                                    116e6a
       BEGIN                                                     116e6a1
       [addr] (rec);                                             116e6a2
       IF addr _ def,dfterm THEN                                 116e6a3
          INVOKE (ptrmrec);                                      116e6a3a
       END;                                                      116e6a4
%lock record%                                                    116e7
    lckrec (rec, ccreate);                                       116e7a
    INVOKE (punlckrec);                                          116e7b
%check for drain%                                                116e8
    IF fld,fddrain THEN sigerr (efodrn);                         116e8a
%add record to folder%                                           116e9
    IF intrec,rcchnb _ brec _ (fld,fdchnb := intrec) THEN        116e9a
```

```
            brec,rcchnf _ intrec;                                    116e9a1
         IF NOT (fld,fdreccnt := fld,fdreccnt + 1) THEN              116e9b
            fld,fdchnf _ intrec;                                     116e9b1
      %return%                                                       116e10
         RETURN (rec, rech);                                         116e10a
         END,                                                        116e10b

(delrec) %delete record%
PROCEDURE (rec POINTER);                                             116f
   %declarations%                                                    116f1
      LOCAL addr;                                                    116f1a
      LOCAL                                                          116f1b
         intrec POINTER, recf POINTER, recb POINTER, def
         POINTER, fld POINTER;                                       116f1b1
   %queue deletion if necessary%                                     116f2
      intrec _ rec - crcofst;                                        116f2a
      fld _ intrec,rcfld;                                            116f2b
      IF NOT intrec,rcdeleted THEN                                   116f2c
         BEGIN                                                       116f2c1
         lckrec (rec, cdelete);                                      116f2c2
         IF fld,fduse THEN                                           116f2c3
            BEGIN                                                    116f2c3a
            intrec,rcdeleted _ TRUE;                                 116f2c3b
            unlckrec (rec, ckeep);                                   116f2c3c
            RETURN;                                                  116f2c3d
            END;                                                     116f2c3e
```

```
        END;                                                        116f2c4

    %remove record from folder%                                     116f3

        recf _ intrec.rcchnf;                                       116f3a

        recb _ intrec.rcchnb;                                       116f3b

        IF recb THEN recb.rcchnf _ recf ELSE fld.fdchnf _
        recf;                                                       116f3c

        IF recf THEN recf.rcchnb _ recb ELSE fld.fdchnb _
        recb;                                                       116f3d

        BUMP DOWN fld.fdreccnt;                                     116f3e

    %flush lock from system%                                        116f4

        IF NOT intrec.rcdeleted THEN unlckrec (rec, ckeep);         116f4a

        trmlcb (intrec.rclcb);                                      116f4b

    %perform user termination%                                      116f5

        def _ fld.fddef;                                            116f5a

        IF addr _ def.dfterm THEN [addr] (rec);                     116f5b

    %release handle%                                                116f6

        orelbit (                                                   116f6a

            fld+fldr.SIZE, def.dfmapl,
            intrec.rchnd-def.dfmnhd+1);                             116f6a1

    %release record%                                                116f7

        relent (cblock, intrec);                                    116f7a

    %return%                                                        116f8

        RETURN;                                                     116f8a

        END.
                                                                    116f8b

(lckrec) %lock record%
PROCEDURE (rec POINTER, type);                                      116g
```

```
        %declarations%                                              116g1
           LOCAL inttype;                                           116g1a
           LOCAL intrec POINTER;                                    116g1b
        %NOP if lock set folder%                                    116g2
           intrec _ rec - crcofst;                                  116g2a
           IF intrec.rcfld = vlsfld THEN RETURN;                    116g2b
        %type CREATE, DELETE = EXCLUSIVE for now%                   116g3
           inttype _ CASE type OF                                   116g3a
              = ccreate, = cdelete: cexclusive;                     116g3a1
              ENDCASE type;                                         116g3a2
        %lock record%                                               116g4
           pshlcb (intrec,rclcb, inttype, FALSE);                   116g4a
        %return%                                                    116g5
           RETURN;                                                  116g5a
           END.
                                                                    116g5b

(unlckrec) %unlock record%
PROCEDURE (rec POINTER, disp);                                      116h
      %delete record%                                               116h1
           IF disp = cdelete THEN delrec (rec);                     116h1a
      %unlock record%                                               116h2
           IF disp AND (rec-crcofst).rcfld # vlsfld THEN            116h2a
              poplcb ();                                            116h2a1
      %return%                                                      116h3
           RETURN;                                                  116h3a
```

```
        END,                                                    1i6h3b

    (chgrec) %change record lock%
    PROCEDURE (rec POINTER, type);                              1i6i

      %declarations%                                            1i6i1

        LOCAL inttype;                                          1i6i1a

      %type CREATE, DELETE = EXCLUSIVE for now%                 1i6i2

        inttype _ CASE type OF                                  1i6i2a

          = ccreate, = cdelete: cexclusive;                     1i6i2a1

          ENDCASE type;                                         1i6i2a2

      %swap lock settings%                                      1i6i3

        swplcb ([rec-crcofst],rclcb, inttype);                  1i6i3a

      %return%                                                  1i6i4

        RETURN;                                                 1i6i4a

        END,                                                    1i6i4b

    (fndrec) %locate record%
    PROCEDURE (fld POINTER, rech, type %-> rec POINTER, rech%); 1i6j

      %declarations%                                            1i6j1

        LOCAL port, hnd, abshnd, addr;                          1i6j1a

        LOCAL rec POINTER, def POINTER;                         1i6j1b

      %make handle absolute%                                    1i6j2

        def _ fld,fddef;                                        1i6j2a

        abshnd _ IF addr _ def,dfhand                           1i6j2b

          THEN [addr] (rech)                                    1i6j2b1

          ELSE rech;                                            1i6j2b2

      %run folder%                                              1i6j3
```

```
            OPENPORT runfld (fld, type : [port]);                    1i6j3a
            WHILE rec _ PCALL [port] (cprevue : hnd) DO              1i6j3b
               IF hnd = abshnd AND PCALL [port] (cuse) THEN          1i6j3b1
                  RETURN (rec, hnd);                                 1i6j3b1a
         %return%                                                    1i6j4
            sigerr (def,dfennex);                                    1i6j4a
            END,
                                                                     1i6j4b

      (rdrec) %retrieve information about record%
      PROCEDURE (type, rec POINTER %-> value%);                      1i6k
         %declarations%                                              1i6k1
            LOCAL value;                                             1i6k1a
            LOCAL pcrid REF;                                         1i6k1b
            LOCAL intrec POINTER;                                    1i6k1c
         %locate requested information%                              1i6k2
            intrec _ rec - crcofst;                                  1i6k2a
            &pcrid _ intrec,rcpcrid;                                 1i6k2b
            CASE type OF                                             1i6k2c
               = cph:   value _ pcrid;                               1i6k2c1
               = cpcrh: value _ pcrid [2];                           1i6k2c2
               ENDCASE  sigerr (efuift);                             1i6k2c3
         %return%                                                    1i6k3
            RETURN (value);                                          1i6k3a
            END,
                                                                     1i6k3b

   %lock management%                                                 1i7
```

```
(inilcb) %initialize LCB%                                          117a
PROCEDURE (lcb POINTER);

   %initialize processor id addr%                                  117a1

      lcb.lcpcrid _ lcb + lcbr.SIZE;                               117a1a

   %return%                                                        117a2

      RETURN;                                                      117a2a

      END,                                                         117a2b

(trmlcb) %terminate LCB%                                           117b
PROCEDURE (lcb POINTER);

   %declarations%                                                  117b1

      LOCAL port, i;                                               117b1a

      LOCAL lckid REF;                                             117b1b

      LOCAL ls POINTER;                                            117b1c

   %purge LCB from folder%                                         117b2

      OPENPORT runfld (vlsfld, FALSE : [port]);                    117b2a

      WHILE ls _ PCALL [port] (clock) DO                           117b2b

         IF (&lckid _ ls.lslckid) AND lckid = lcb THEN             117b2b1

            BEGIN                                                  117b2b1a

            IF NOT ls.lslocked THEN                                117b2b1b

               sigecb (ls.lsecb, elfdel);                          117b2b1b1

            delrec (ls);                                           117b2b1c

            END;                                                   117b2b1d

   %purge LCB from stack%                                          117b3

      FOR i _ 1 UP UNTIL > vlckstk DO                              117b3a

         IF vlckstk [i].ldlcb = lcb THEN                           117b3a1
```

```
            vlckstk [i],lddead _ TRUE;                          1i7b3a1a
    %return%                                                    1i7b4
        RETURN;                                                 1i7b4a
        END,
                                                                1i7b4b
(pshlcb) %push LCB lock%
PROCEDURE (lcb POINTER, type, handleornot);                     1i7c
    %declarations%                                              1i7c1
        LOCAL ldsc, ecb;                                        1i7c1a
        LOCAL lckid = (lcb, type, cprocessor);                  1i7c1b
    %allocate ECB%                                              1i7c2
        ecb _ aloecb (1);                                       1i7c2a
        INVOKE (palwrb,, ecb);                                  1i7c2b
    %lock LCB%                                                  1i7c3
        ldsc _ setlcb ($lckid, ecb, handleornot);               1i7c3a
        waicok (ecb);                                           1i7c3b
    %check for stack overflow%                                  1i7c4
        IF vlckstk >= cmlkskl THEN sigerr (elostk);             1i7c4a
    %push setting onto stack%                                   1i7c5
        vlckstk [vlckstk _ vlckstk + 1] _ ldsc;                 1i7c5a
    %return%                                                    1i7c6
        RETURN;                                                 1i7c6a
        END,
                                                                1i7c6b
(poplcb) %pop LCB lock%
PROCEDURE;                                                      1i7d
    %check for stack underflow%                                 1i7d1
```

```
        IF vlckstk <= 0 THEN sigerr (elmstk);                    1i7d1a
      %pop setting off of stack%                                 1i7d2
        remlcb (vlckstk [vlckstk := vlckstk - 1]);               1i7d2a
      %return%                                                   1i7d3
        RETURN;                                                  1i7d3a
        END,                                                     1i7d3b

(swplcb) %swap LCB lock%
PROCEDURE (lcb POINTER, type);                                   1i7e
    %declarations%                                               1i7e1
      LOCAL ldsc, i, ecb;                                        1i7e1a
      LOCAL lckid = (lcb, type, cprocessor);                     1i7e1b
    %locate affected stack position%                             1i7e2
      LOOP                                                       1i7e2a
        BEGIN                                                    1i7e2a1
        FOR i _ vlckstk DOWN UNTIL < 1 DO                        1i7e2a2
          IF vlckstk [i].ldlcb = lcb THEN EXIT LOOP 2;           1i7e2a2a
        sigerr (elmswp);                                         1i7e2a3
        END;                                                     1i7e2a4
    %allocate ECB%                                               1i7e3
      ecb _ aloecb (1);                                          1i7e3a
      INVOKE (palwrb,, ecb);                                     1i7e3b
    %instate new lock setting%                                   1i7e4
      ldsc _ setlcb (slckid, ecb, FALSE);                        1i7e4a
      waicok (ecb);                                              1i7e4b
    %remove old setting%                                         1i7e5
```

```
        remlcb (vlckstk [i] := ldsc);                                   1i7e5a
    %return%                                                            1i7e6
        RETURN;                                                         1i7e6a
        END,                                                            1i7e6b

(setlcb) %lock LCB%
PROCEDURE (lckid REF, ecb REF, handleornot %-> ldsc%);                  1i7f
    %declarations%                                                      1i7f1
        LOCAL ldsc, type;                                               1i7f1a
        LOCAL pcrid = (0, 0, vmgh);                                     1i7f1b
        LOCAL impcrid REF;                                              1i7f1c
        LOCAL ls POINTER, lcb POINTER;                                  1i7f1d
    %construct processor id%                                            1i7f2
        IF handleornot THEN oblkxfr ($vcpcrid, $pcrid,
        cmpidl);                                                        1i7f2a
    %record lock in LCB if possible%                                    1i7f3
        ldsc        _ lcb _ lckid;                                      1i7f3a
        ldsc.ldtype _ type _ lckid [1];                                 1i7f3b
        &impcrid _ lcb.lcpcrid;                                         1i7f3c
        IF NOT handleornot                                              1i7f3d
        AND (type = cshare OR type = cexclusive)                        1i7f3e
        AND (NOT (lcb.lcimex OR lcb.lcimsh)                             1i7f3f
           OR impcrid [2] = pcrid [2])                                  1i7f3f1
        AND NOT lcb.lcfdex                                              1i7f3g
        AND NOT (type = cexclusive AND lcb.lcfdsh)                      1i7f3h
        THEN                                                            1i7f3i
```

```
        BEGIN                                                   1i7f3i1
        %advance meter%                                         1i7f3i2
           BUMP vmflkc;                                         1i7f3i2a
        %save processor id%                                     1i7f3i3
           IF NOT impcrid THEN                                  1i7f3i3a
              oblkxfr (spcrid, &impcrid, cmpidl);               1i7f3i3a1
        %bump appropriate lock count%                           1i7f3i4
           IF type = cexclusive                                 1i7f3i4a
              THEN BUMP lcb.lcimex                              1i7f3i4a1
              ELSE BUMP lcb.lcimsh;                             1i7f3i4a2
        %signal lock set%                                       1i7f3i5
           sigecb (&ecb, cok);                                  1i7f3i5a
        %return%                                                1i7f3i6
           RETURN (ldsc);                                       1i7f3i6a
        END;                                                    1i7f3i7
     %advance meter%                                            1i7f4
        BUMP vmslkc;                                            1i7f4a
     %create lockset record%                                    1i7f5
        ls _ crtrec (vlsfld, call : ldsc,ldlsh);                1i7f5a
        INVOKE (prtnkp,, ls);                                   1i7f5b
        oblkxfr (&lckid, ls.lslckid, cmlkidl);                  1i7f5c
        oblkxfr (spcrid, ls.lspcrid, cmpidl);                   1i7f5d
        ls.lsecb    _ &ecb;                                     1i7f5e
        ls.lshandle _ handleornot;                              1i7f5f
     %set lock if possible%                                     1i7f6
```

```
        IF okwall (&lckid, spcrid) AND ls.lslocked _ TRUE THEN 1i7f6a
           BEGIN                                               1i7f6a1
           CASE type OF                                        1i7f6a2
              = cshare:     BUMP lcb.lcfdsh;                   1i7f6a2a
              = cexclusive: BUMP lcb.lcfdex;                   1i7f6a2b
              ENDCASE       BUMP lcb.lcfdsp;                   1i7f6a2c
           IF &ecb THEN sigecb (&ecb, cok);                    1i7f6a3
           END                                                 1i7f6a4
        ELSE                                                   1i7f6b
           IF &ecb THEN                                        1i7f6b1
              BEGIN                                            1i7f6b1a
              %advance meter%                                  1i7f6b1b
                 BUMP vmwlkc;                                  1i7f6b1b1
              %trace waited manager%                           1i7f6b1c
                 IF dtrace ,A ctrlkwt THEN trcmg (chold,
                 vmgh);                                        1i7f6b1c1
              %assure no deadlock%                             1i7f6b1d
                 nodedlck (&lckid, spcrid, 0, 0);              1i7f6b1d1
              %bump queued request count%                      1i7f6b1e
                 BUMP lcb.lcfdrq                               1i7f6b1e1
              END                                              1i7f6b1f
           ELSE sigerr (elflck);                               1i7f6b2
     %return%                                                  1i7f7
        RETURN (ldsc);                                         1i7f7a
        END.
                                                               1i7f7b
```

```
(remlcb) %unlock LCB%
PROCEDURE (ldsc);                                                    1i7g

   %declarations%                                                    1i7g1

      LOCAL type, lsh, pcrid, port, statusquo=FALSE;                 1i7g1a

      LOCAL wklckid [cmlkidl];                                       1i7g1b

      LOCAL qlckid REF, lckid REF, qpcrid REF;                       1i7g1c

      LOCAL ls POINTER, lcb POINTER;                                 1i7g1d

   %catchphrases%                                                    1i7g2

      (pdells) CATCHPHRASE; IF rtn () THEN delrec (ls);              1i7g2a

   %NOP if descriptor dead%                                          1i7g3

      IF ldsc.lddead THEN RETURN;                                    1i7g3a

   %lock recorded in LCB%                                            1i7g4

      lcb _ ldsc.ldlcb;                                              1i7g4a

      type _ ldsc.ldtype;                                            1i7g4b

      IF NOT lsh _ ldsc.ldlsh THEN                                   1i7g4c

         BEGIN                                                       1i7g4c1

         statusquo _ CASE type OF                                    1i7g4c2

            = cshare: lcb.lcimsh _ lcb.lcimsh - 1;                   1i7g4c2a

            ENDCASE   lcb.lcimex _ lcb.lcimex - 1;                   1i7g4c2b

         IF NOT statusquo THEN                                       1i7g4c3

            BEGIN                                                    1i7g4c3a

            pcrid _ $vcpcrid;                                        1i7g4c3b

            wklckid     _ lcb;                                       1i7g4c3c

            wklckid [1] _ type;                                      1i7g4c3d

            wklckid [2] _ cprocessor;                                1i7g4c3e
```

```
            &lckid _ swklckid;                                  1i7g4c3f
            END;                                                1i7g4c3g
         END                                                    1i7g4c4
      %lock recorded in folder%                                 1i7g5
         ELSE                                                   1i7g5a
            BEGIN                                               1i7g5a1
            ls _ fndrec (vlsfld, lsh, FALSE);                   1i7g5a2
            INVOKE (pdells);                                    1i7g5a3
            &lckid _ ls.lslckid;                                1i7g5a4
            pcrid _ ls.lspcrid;                                 1i7g5a5
            IF NOT statusquo _ NOT (ls.lslocked := FALSE) THEN  1i7g5a6
               CASE lckid [1] OF                                1i7g5a6a
                  = cshare:     BUMP DOWN lcb.lcfdsh;           1i7g5a6a1
                  = cexclusive: BUMP DOWN lcb.lcfdex;           1i7g5a6a2
                  ENDCASE       BUMP DOWN lcb.lcfdsp;           1i7g5a6a3
            END;                                                1i7g5a7
      %reevaluate queued lock requests%                         1i7g6
         IF NOT statusquo AND lcb.lcfdrq THEN                   1i7g6a
            BEGIN                                               1i7g6a1
            OPENPORT runfld (vlsfld, FALSE : [port]);           1i7g6a2
            WHILE ls _ PCALL [port] (clock) DO                  1i7g6a3
               IF NOT ls.lslocked                               1i7g6a3a
               AND NOT                                          1i7g6a3b
                  okwone (&lckid, pcrid, &qlckid _ ls.lslckid,
                  &qpcrid _ ls.lspcrid)                         1i7g6a3b1
```

```
                AND okwall (&qlckid, &qpcrid) THEN                  1i7g6a3c
                   BEGIN                                            1i7g6a3c1
                   %update queued and locked counts%                1i7g6a3c2
                      ls,lslocked _ TRUE;                           1i7g6a3c2a
                      CASE qlckid [1] OF                            1i7g6a3c2b
                         = cshare:     BUMP lcb,lcfdsh;             1i7g6a3c2b1
                         = cexclusive: BUMP lcb,lcfdex;             1i7g6a3c2b2
                         ENDCASE       BUMP lcb,lcfdsp;             1i7g6a3c2b3
                      BUMP DOWN lcb,lcfdrq;                         1i7g6a3c2c
                   %notify manager%                                 1i7g6a3c3
                      sigecb (ls,lsecb, cok);                       1i7g6a3c3a
                   %trace released manager%                         1i7g6a3c4
                      IF dtrace ,A ctrlkwt THEN                     1i7g6a3c4a
                         trcmg (crelease, qpcrid [2]);              1i7g6a3c4a1
                   END;                                             1i7g6a3c5
             END;                                                   1i7g6a4
          %return%                                                  1i7g7
             RETURN;                                                1i7g7a
             END,
                                                                    1i7g7b
       %event management%                                           1i8

          (aloecb) %allocate ECB%
          PROCEDURE (length %-> ecb REF%);                          1i8a
             %declarations%                                         1i8a1
                LOCAL ecb REF;                                      1i8a1a
             %allocate ECB%                                         1i8a2
```

```
        &ecb _ aloent (cblock, 1+length);                    1i8a2a
    %initialize ECB%                                         1i8a3
        ecb.M _ length;                                      1i8a3a
    %return%                                                 1i8a4
        RETURN (&ecb);                                       1i8a4a
        END.                                                 1i8a4b

(sigecb) %signal event%
PROCEDURE (ecb REF, code);                                   1i8b
    %advance meter%                                          1i8b1
        BUMP vmsigc;                                         1i8b1a
    %check for ECB overflow%                                 1i8b2
        IF ecb.L = ecb.M THEN sigerr (evoecb);               1i8b2a
    %append new code to ECB%                                 1i8b3
        ecb [ecb.L _ ecb.L + 1] _ code;                      1i8b3a
    %return%                                                 1i8b4
        RETURN;                                              1i8b4a
        END.                                                 1i8b4b

(rececb) %recall signal%
PROCEDURE (ecb REF, code);                                   1i8c
    %declarations%                                           1i8c1
        LOCAL i;                                             1i8c1a
    %locate code and delete from ECB%                        1i8c2
        FOR i _ ecb.L DOWN UNTIL < 1 DO                      1i8c2a
            IF ecb [i] = code THEN                           1i8c2a1
```

```
            oblkxfr (&ecb+i+1, &ecb+i, (ecb.L := ecb.L - 1)
            - i);                                              1i8c2a1a

      %return%                                                 1i8c3

         RETURN;                                               1i8c3a

         END.                                                  1i8c3b

(waiecb) %wait for signal%
PROCEDURE (ecb REF %-> code%);                                 1i8d

      %declarations%                                           1i8d1

         LOCAL code;                                           1i8d1a

      %advance meter%                                          1i8d2

         BUMP vmvwtc;                                          1i8d2a

      %wait for signal%                                        1i8d3

         IF NOT (code _ otest (&ecb, TRUE)) THEN               1i8d3a

            code _ owait (&ecb, FALSE);                        1i8d3a1

      %return%                                                 1i8d4

         RETURN (code);                                        1i8d4a

         END.                                                  1i8d4b

(waicok) %wait for signal COK%
PROCEDURE (ecb REF);                                           1i8e

      %declarations%                                           1i8e1

         LOCAL code;                                           1i8e1a

      %wait for signal%                                        1i8e2

         IF &ecb AND (code _ waiecb (&ecb)) # cok THEN         1i8e2a

            sigerr (code);                                     1i8e2a1

      %return%                                                 1i8e3
```

```
        RETURN;                                                 118e3a

        END,
                                                                118e3b

%data structure conversion%                                     119

   (encprms) %encode parameters according to specs%
   PROCEDURE (prm1 REF, cnt, specs REF, dst REF %-> prm1 REF%);  119a

      %declarations%                                            119a1

         LOCAL bp, i;                                           119a1a

         LOCAL intdst REF;                                      119a1b

      %verify number of parameters%                             119a2

         IF prm1.L # cnt THEN sigerr (edwpmc);                  119a2a

      %select destination%                                      119a3

         &intdst _ IF &dst THEN &dst ELSE &prm1;                119a3a

      %loop through list of parameters%                         119a4

         bp _ opntwrd (&specs, cprpmspcl);                      119a4a

         FOR i _ 1 UP UNTIL > cnt DO                            119a4b

            #intdst# [i] _                                      119a4b1

               USE descrb (encstruc ((ordbyt ($bp)).spdtype,
               ELEM #prm1# [i]));                               119a4b1a

      %return%                                                  119a5

         RETURN (&intdst);                                      119a5a

         END,
                                                                119a5b

   (decprms) %decode parameters according to specs%
   PROCEDURE (prm1 REF, cnt, specs REF %-> prm1 REF%);          119b

      %declarations%                                            119b1

         LOCAL bp, i;                                           119b1a
```

```
        %verify number of parameters%                                   1i9b2

           IF prml,L # cnt THEN  sigerr (edwpmc);                       1i9b2a

        %loop through list of data structures%                          1i9b3

           bp _ opntwrd (&specs, cprpmspcl);                            1i9b3a

           FOR i _ 1 UP UNTIL > cnt DO                                  1i9b3b

              #prml# [i] _                                              1i9b3b1

                 USE decstruc ((ordbyt (sbp)),spdtype, ELEM
                 #prml# [i], -i);                                       1i9b3b1a

        %return%                                                        1i9b4

           RETURN (&prml);                                              1i9b4a

           END,                                                         1i9b4b

     (encstruc) %encode data structure%
     PROCEDURE (type, src REF %-> dst REF%);                            1i9c

        %declarations%                                                  1i9c1

           LOCAL                                                        1i9c1a

              intsrc, i, ub, funtype=type.tyfunt,
              listof=type.tylist;                                       1i9c1a1

           LOCAL dst REF;                                               1i9c1b

        %EMPTY OK%                                                      1i9c2

           IF type,tyempok AND NOT &src THEN                            1i9c2a

              BEGIN                                                     1i9c2a1

              funtype _ cempty;                                         1i9c2a2

              listof _ FALSE;                                           1i9c2a3

              END                                                       1i9c2a4

        %eliminate synonyms%                                            1i9c3

           ELSE funtype _ CASE funtype OF                               1i9c3a
```

```
            = cucstr: ccharstr;                                    1i9c3a1
            = cdumemp: cempty;                                     1i9c3a2
            = cdumlist: clist;                                     1i9c3a3
            = cdata, = calodata, = calostruc: cstruc               1i9c3a4
            ENDCASE funtype;                                       1i9c3a5
        %encode elements if list%                                  1i9c4
          IF listof THEN                                           1i9c4a
            BEGIN                                                  1i9c4a1
            intsrc _ &dst _ aloent (clist, ub _ src,L);            1i9c4a2
            INVOKE (palwrl,, &dst);                                1i9c4a3
            FOR i _ 1 UP UNTIL > ub DO                             1i9c4a4
              #dst# !_                                             1i9c4a4a
                USE descrb (encstruc (funtype, ELEM #src#
                [i]));                                             1i9c4a4a1
            funtype _ clist;                                       1i9c4a5
            END                                                    1i9c4a6
          ELSE intsrc _ &src;                                      1i9c4b
        %return%                                                   1i9c5
          RETURN (                                                 1i9c5a
            [IF funtype IN [1, cmfdtn] THEN sencfstruc ELSE
            sencistruc] (funtype, intsrc));                        1i9c5a1
          END,
                                                                   1i9c5b
      (decstruc) %decode data structure%
      PROCEDURE (type, src REF, optdst REF %-> dst REF%);          1i9d
        %declarations%                                             1i9d1
          LOCAL ub, i, funtype=type,tyfunt, listof=type,tylist;    1i9d1a
```

```
    LOCAL dst REF, dstaddr REF =0;                                  1i9d1b
%EMPTY OK%                                                          1i9d2
    IF type,tyempok AND src,dstype = cempty THEN                    1i9d2a
      BEGIN                                                         1i9d2a1
      funtype _ cempty;                                             1i9d2a2
      listof _ FALSE;                                               1i9d2a3
      END                                                           1i9d2a4
%eliminate synonyms%                                                1i9d3
    ELSE funtype _ CASE funtype OF                                  1i9d3a
      = cdumemp: cempty;                                            1i9d3a1
      = cdumlist: clist;                                            1i9d3a2
      = cdata, = calodata, = calostruc: cstruc                      1i9d3a3
      ENDCASE funtype;                                              1i9d3a4
%decode elements if list%                                           1i9d4
    IF listof THEN                                                  1i9d4a
      BEGIN                                                         1i9d4a1
      &dstaddr,RH _ &dst _                                          1i9d4a2
        decstruc (clist, &src, &optdst);                            1i9d4a2a
      IF &dstaddr # &optdst THEN INVOKE (pabrr1,,
      &dstaddr);                                                    1i9d4a3
      ub _ dstaddr,L;                                               1i9d4a4
      FOR i _ 1 UP UNTIL > ub DO                                    1i9d4a5
        #dstaddr# [i] _                                             1i9d4a5a
          USE decstruc (funtype, ELEM #dstaddr# [i],
          -1);                                                      1i9d4a5a1
      END                                                           1i9d4a6
```

```
    %decode structure%                                          119d5

       ELSE                                                     119d5a

          &dst _ [IF funtype IN [1, cmfdtn] THEN $decfstruc
          ELSE $decistruc] (funtype, &src, &optdst);            119d5a1

    %return%                                                    119d6

       RETURN (&dst);                                           119d6a

       END,                                                     119d6b

(deckey) %decode data structure key%
PROCEDURE (type, src REF, optdst REF %-> dst REF%);             119e

    %declarations%                                              119e1

       LOCAL offset;                                            119e1a

    %verify presence of key%                                    119e2

       IF NOT src.dskey THEN sigerr (edmkey);                   119e2a

       sizstruc (&src ; offset);                                119e2b

    %return%                                                    119e3

       RETURN (decstruc (type, &src + offset, &optdst));        119e3a

       END,                                                     119e3b

(addkey) %combine key and data structure%
PROCEDURE (key REF, src REF %-> dst REF%);                      119f

    %declarations%                                              119f1

       LOCAL keylen, srclen;                                    119f1a

       LOCAL dst REF;                                           119f1b

    %verify key status%                                         119f2

       IF src.dskey THEN sigerr (eddkey);                       119f2a

       IF key.dskey THEN sigerr (eddrky);                       119f2b
```

```
        %allocate space for structure%                                  1i9f3
           &dst _                                                       1i9f3a
              aloent (cblock,                                           1i9f3a1
                 (keylen _ sizent (cblock, &key)) +                     1i9f3a1a
                 (srclen _ sizent (cblock, &src)));                     1i9f3a1b
           INVOKE (pabrrb,, &dst);                                      1i9f3b
        %combine key and src%                                           1i9f4
           oblkxfr (&src, &dst, srclen);                                1i9f4a
           oblkxfr (&key, &dst+srclen, keylen);                         1i9f4b
           dst.dskey _ TRUE;                                            1i9f4c
        %return%                                                        1i9f5
           RETURN (&dst);                                               1i9f5a
           END.                                                         1i9f5b

     (savstruc) %save data structure%
     PROCEDURE (src REF %=> dst REF%);                                  1i9g
        %declarations%                                                  1i9g1
           LOCAL len;                                                   1i9g1a
           LOCAL dst REF;                                               1i9g1b
        %allocate space for structure%                                  1i9g2
           &dst _ aloent (cblock, len _ sizstruc (&src));               1i9g2a
           INVOKE (pabrrb,, &dst);                                      1i9g2b
        %copy structure%                                                1i9g3
           oblkxfr (&src, &dst, len);                                   1i9g3a
        %return%                                                        1i9g4
           RETURN (&dst);                                               1i9g4a
```

```
        END,                                                    1i9g4b

    (cpestruc) %compare data structures%
    PROCEDURE (struc1 POINTER, struc2 POINTER %-> outcome%);    1i9h
      %declarations%                                            1i9h1
        LOCAL key1, key2, type, len, len2, elem1, elem2, i;     1i9h1a
      %catchphrases%                                            1i9h2
        (prstkfg) CATCHPHRASE; IF abrtn () THEN                 1i9h2a
          BEGIN                                                 1i9h2a1
          struc1.dskey _ key1;                                  1i9h2a2
          struc2.dskey _ key2;                                  1i9h2a3
          END;                                                  1i9h2a4
      %verify like types%                                       1i9h3
        IF (type _ struc1.dstype) # struc2.dstype THEN          1i9h3a
          RETURN (FALSE);                                       1i9h3a1
      %both lists%                                              1i9h4
        IF type = clist THEN                                    1i9h4a
          BEGIN                                                 1i9h4a1
          %verify same no. elements in each%                    1i9h4a2
            IF (len _ struc1.dslen) # struc2.dslen THEN         1i9h4a2a
              RETURN (FALSE);                                   1i9h4a2a1
          %compare corresponding elements%                      1i9h4a3
            elem1 _ struc1 + 1;                                 1i9h4a3a
            elem2 _ struc2 + 1;                                 1i9h4a3b
            FOR i _ 1 UP UNTIL > len DO                         1i9h4a3c
              IF NOT cpestruc (                                 1i9h4a3c1
```

```
                    elem1 := elem1 + sizstruc (elem1),      1i9h4a3c1a
                    elem2 := elem2 + sizstruc (elem2)) THEN
                                                            1i9h4a3c1b
                       RETURN (FALSE);                      1i9h4a3c1b1
        END                                                 1i9h4a4
    %both non-lists%                                        1i9h5
      ELSE                                                  1i9h5a
        BEGIN                                               1i9h5a1
        %verify same length%                                1i9h5a2
          sizstruc (struc1 :  len);                         1i9h5a2a
          sizstruc (struc2 : len2);                         1i9h5a2b
          IF len # len2 THEN RETURN (FALSE);                1i9h5a2c
        %keys irrelevant -- clear both key flags%           1i9h5a3
          key1 _ (struc1.dskey := FALSE);                   1i9h5a3a
          key2 _ (struc2.dskey := FALSE);                   1i9h5a3b
          INVOKE (prstkfg);                                 1i9h5a3c
        %verify same value%                                 1i9h5a4
          IF NOT blkcpe (struc1, struc2, len)               1i9h5a4a
            THEN RETURN (FALSE);                            1i9h5a4a1
        END;                                                1i9h5a5
    %return%                                                1i9h6
      RETURN (TRUE);                                        1i9h6a
      END,
                                                            1i9h6b
(sizstruc) %compute size of data structure%
PROCEDURE (src POINTER %-> blksize, nokeysize%);            1i9i
  %declarations%                                            1i9i1
```

```
    LOCAL len, blksize, nxtsize, nxtsrc, i, nokeysize;          1i911a
  %dispatch on basis of data type%                              1i912
    len _ src.dslen;                                            1i912a
    CASE src.dstype OF                                          1i912b
      = cempty, = cboolean, = cindex: blksize _ 1;              1i912b1
      = cinteger: blksize _ 2;                                  1i912b2
      = cbitstr:  blksize _ 1 + (len+35)/36;                    1i912b3
      = ccharstr: blksize _ 1 + (len+4)/5;                      1i912b4
      = clist:                                                  1i912b5
        BEGIN                                                   1i912b5a
        blksize _ nxtsize _ 1;                                  1i912b5b
        nxtsrc _ src;                                           1i912b5c
        FOR i _ 1 UP UNTIL > len DO                             1i912b5d
          blksize _ blksize + (nxtsize _                        1i912b5d1
            sizstruc (nxtsrc _ nxtsrc + nxtsize));              1i912b5d1a
        END;                                                    1i912b5e
      ENDCASE sigerr (edufty);                                  1i912b6
  %include key if any%                                          1i913
    nokeysize _ blksize;                                        1i913a
    IF src.dskey THEN                                           1i913b
      blksize _ nokeysize + sizstruc (src + blksize);           1i913b1
  %return%                                                      1i914
    RETURN (blksize, nokeysize);                                1i914a
    END;
                                                                1i914b
```

```
(fndelm) %locate element of externally-formatted list%
PROCEDURE (list REF, esel REF %-> elem REF, offset%);                    1i9j

   %declarations%                                                        1i9j1

      LOCAL i, j, len, eselelm, keyoffset, offset, index,
      ub;                                                               1i9j1a

      LOCAL src POINTER;                                                1i9j1b

   %descend into structure via element selector%                        1i9j2

      ub _ IF &esel THEN esel,L ELSE 0;                                 1i9j2a

      src _ &list;                                                      1i9j2b

      FOR i _ 1 UP UNTIL > ub DO                                        1i9j2c

         BEGIN                                                          1i9j2c1

         %verify structure a list%                                      1i9j2c2

            IF src,dstype # clist THEN sigerr (edwesl);                 1i9j2c2a

            len _ src,dslen;                                            1i9j2c2b

            BUMP src;                                                   1i9j2c2c

         %search by key%                                                1i9j2c3

            IF deckey (cboolean, eselelm _ ELEM #esel# [i])
            THEN                                                        1i9j2c3a

               BEGIN                                                    1i9j2c3a1

               FOR j _ 1 UP UNTIL > len DO                              1i9j2c3a2

                  IF src,dskey THEN                                     1i9j2c3a2a

                     BEGIN                                              1i9j2c3a2a1

                     offset _ sizstruc (src : keyoffset);
                                                                        1i9j2c3a2a2

                     IF cpestruc (eselelm, src + keyoffset)
                     THEN                                               1i9j2c3a2a3

                        REPEAT LOOP 2;                                  1i9j2c3a2a3a

                     src _ src + offset;                                1i9j2c3a2a4
```

```
                    END;                                        1i9j2c3a2a5
                sigerr (edwkey);                                1i9j2c3a3
                END                                             1i9j2c3a4
            %search by index%                                   1i9j2c4
              ELSE                                              1i9j2c4a
                BEGIN                                           1i9j2c4a1
                %verify target index%                           1i9j2c4a2
                  IF (index _ decstruc (cindex, eselelm, 0))
                  > len THEN                                    1i9j2c4a2a
                    sigerr (edwidx);                            1i9j2c4a2a1
                %skip to element%                               1i9j2c4a3
                  ub _ index - 1;                               1i9j2c4a3a
                  FOR j _ 1 UP UNTIL > ub DO                    1i9j2c4a3b
                    src _ src + sizstruc (src);                 1i9j2c4a3b1
                END;                                            1i9j2c4a4
            END;                                                1i9j2c5
        %return%                                                1i9j3
          RETURN (src, src-&list);                              1i9j3a
          END,
                                                                1i9j3b
    %storage management%                                        1i10
      (alcent) %allocate entity%
      PROCEDURE (type, length %-> ent REF%);                    1i10a
        %declarations%                                          1i10a1
          LOCAL addr, ent;                                      1i10a1a
        %verify size of entity%                                 1i10a2
```

```
        IF length < 0                                                   1110a2a
        OR (type = cblock AND NOT length) THEN                          1110a2b
            sigerr (emient);                                            1110a2b1
    %select allocation routine%                                         1110a3
        addr _ CASE type OF                                             1110a3a
            = cblock:   $getblk;                                        1110a3a1
            = ccharstr: $getstring;                                     1110a3a2
            = clist:    $getlst;                                        1110a3a3
            ENDCASE sigerr (emuent);                                    1110a3a4
    %allocate entity%                                                   1110a4
        IF NOT ent _ [addr] (length, vstgzon) THEN                      1110a4a
            sigerr (emfexh);                                            1110a4a1
    %return%                                                            1110a5
        RETURN (ent);                                                   1110a5a
        END,                                                            1110a5b

(relent) %release entity%
PROCEDURE (type, ent REF);                                              1110b
    %declarations%                                                      1110b1
        LOCAL addr;                                                     1110b1a
    %NOP if no entity%                                                  1110b2
        IF NOT &ent THEN RETURN;                                        1110b2a
    %select deallocation routine%                                       1110b3
        CASE type OF                                                    1110b3a
            = cblock, = ccharstr: addr _ $freeblk;                      1110b3a1
            = clist:                                                    1110b3a2
```

```
                BEGIN                                                   1i10b3a2a
                #ent# _;                                                1i10b3a2b
                addr _ $frelst;                                         1i10b3a2c
                END;                                                    1i10b3a2d
            ENDCASE sigerr (emuent);                                    1i10b3a3
        %release storage%                                               1i10b4
            [addr] ((&ent),RH, vstgzon);                                1i10b4a
        %return%                                                        1i10b5
            RETURN;                                                     1i10b5a
            END,                                                        1i10b5b

    (savent) %save entity%
    PROCEDURE (type, src REF %-> dst REF%);                             1i10c
        %declarations%                                                  1i10c1
            LOCAL len;                                                  1i10c1a
            LOCAL dst REF;                                              1i10c1b
        %catchphrases%                                                  1i10c2
            (prelent) CATCHPHRASE; IF abr () THEN                       1i10c2a
                relent (type, &dst);                                    1i10c2a1
        %NOP if no source entity%                                       1i10c3
            IF NOT &src THEN RETURN (0);                                1i10c3a
        %allocate space for copy of entity%                             1i10c4
            &dst _ aloent (type, len _ sizent (type, &src));            1i10c4a
            INVOKE (prelent);                                           1i10c4b
        %copy entity%                                                   1i10c5
            CASE type OF                                                1i10c5a
```

```
            = cblock:   oblkxfr (&src, &dst, len);                  1110c5a1
            = ccharstr: #dst# _ #src#;                              1110c5a2
            = clist:    #dst# _ COPY #src#;                         1110c5a3
            ENDCASE     sigerr (emuent);                            1110c5a4
        %return%                                                    1110c6
            RETURN (&dst);                                          1110c6a
            END,
                                                                    1110c6b
    (sizent) %compute length of entity%
    PROCEDURE (type, ent REF %=> length%);                          1110d
        %declarations%                                              1110d1
            LOCAL len;                                              1110d1a
        %NOP if no entity%                                          1110d2
            IF NOT &ent THEN RETURN (0);                            1110d2a
        %compute length of entity%                                  1110d3
            len _ CASE type OF                                      1110d3a
                = cblock:                                           1110d3a1
                    ent [= blkhdr.SIZE].blklength = blkhdr.SIZE;    1110d3a1a
                = ccharstr, = clist:                                1110d3a2
                    ent.L;                                          1110d3a2a
                ENDCASE sigerr (emuent);                            1110d3a3
        %return%                                                    1110d4
            RETURN (len);                                           1110d4a
            END,
                                                                    1110d4b
    (blkcpe) %compare blocks%
    PROCEDURE (blk1 REF, blk2 REF, length %=> outcome%);            1110e
```

```
    %declarations%                                              1110e1
      LOCAL i, ub;                                              1110e1a
    %verify length of block%                                    1110e2
      IF length < 0 THEN sigerr (emient);                       1110e2a
    %compare blocks%                                            1110e3
      ub _ length - 1;                                          1110e3a
      FOR i _ 0 UP UNTIL > ub DO                                1110e3b
        IF blk1 [i] # blk2 [i] THEN RETURN (FALSE);             1110e3b1
    %return%                                                    1110e4
      RETURN (TRUE);                                            1110e4a
      END,                                                      1110e4b

(mkarray) %construct array from list%
PROCEDURE (src REF, dst REF, xfrornot);                         1110f
    %declarations%                                              1110f1
      LOCAL i, ub;                                              1110f1a
    %move elements from list to array%                          1110f2
      ub _ src,L;                                               1110f2a
      FOR i _ 1 UP UNTIL > ub DO                                1110f2b
        rdlelm (&src, xfrornot, i : dst [i-1]);                 1110f2b1
    %return%                                                    1110f3
      RETURN;                                                   1110f3a
      END,                                                      1110f3b

(descrb) %construct block list descriptor%
PROCEDURE (src REF %-> descr%);                                 1110g
    %declarations%                                              1110g1
```