DSPGEN

```
< NLS, DSPGEN.NLS.37, >, 14-Dec-77 15:49 JDH ;;;;;
FILE dspgen  %  L10  <rel-nls>dspgen  %% (L10,) (rel-nls,dspgen.rel,) %
                                                                      02
%ask chi or dsm about fixing startup commands that get lda (someone made
comments in dspgen)%                                                06296
%.....Declarations.....%                                              03
   REF msgda, litda, namda, cflda, ltvsda, vspcda, rt, art, mkrptr;   04
   REF lppch; % Line processor printer char routine %                02582
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4, s = 9, m = 10;            05
   DECLARE STRING                                                     06
      mt = "Empty",                                                   07
         %string can be changed without program modification          08
         if less than 20 characters in length%                        09
      dotstr = ".............................",
      unknown = "<UKC>" %unknown char detected in npstrad%;           010
   DECLARE EXTERNAL deltadata = 1, hazeltine = 2, datamedia = 3,
   dm4000=5;                                                          011
      %plan not to use codes 5 and above except to set flags in getdev%
                                                                      08522
   DECLARE  %source codes for LSRT%                                   012
      srcnul = 0, srcstat = 1, srcstno = 2, srcsig = 3, srcast        013
      = 4, srcdot = 5;                                                014
   DECLARE shrink = 4; %how many consecutive times user can           015
      have too much LSRT before system reallocates%                   016
   DECLARE  %display commands%                                        017
      writelsg = 1, movelsg = 2, deletlsg = 3, suppressda = 4, restoreda
      = 5, blankda = 6;                                               018
   SET BLT = 251B;                                                    019
                                                                      020
   DECLARE  %command processing constants%                            021
      forward = 1, backward = 2;                                      022
%.....CORE-NLS.....%
                                                                      023
   %....Display format control....%                                   024
      (dafrmt)   PROCEDURE (da, oldsw); %display area formatter%      025
         %This routine generates the display image for a text display
         area, assuming the universal display.  It is driven entirely by
         the information available in the display area record (da).
         Returns the t-pointer for the next statement character after
         the last one which was displayed.  oldsw is the address of an
         existing sequence work area or zero.  If zero, then a sequence
         is opened and closed in this procedure.%                     026
         %----------------%                                           027
         LOCAL end, stid, charct, sa, ptr, endflg, fmtstd, sv1, sv2,
         frzarray[40];                                                028
         REF da, sa, oldsw;                                           029
         IF NOT da.daexis THEN err($"Fatal display error in DAFRMT");
                                                                      030
         IF NOT da.davspec.vsdaft THEN RETURN; %defer recreate display%
                                                                     07625
            %                                                        07626
            this block former defer recreate code; caused loop when
            inserting more than one line of text with VS v on:       07627
            BEGIN                                                    07628
            IF lplitreset THEN                                       07629
               BEGIN                                                 07630
```

```
        litline _ lplitline;                                    07631
        litreset _ FALSE;                                       07632
        litapflag _ TRUE;                                       07633
        rstlit();                                               07634
        lplitreset _ FALSE;                                     07635
        END;                                                    07636
    IF NOT litreset THEN rstlit();                              07637
    RETURN;                                                     07638
    END;                                                        07639
    %                                                           07640
%..INITIALIZATION....%                                           037
    %..update previous file in da - for jump routines....%       038
        da.dapstf _ da.dacsp.stfile;                             039
    lplitreset _ FALSE;                                         05797
    %..if csp is not valid, set to origin..%                     040
        IF NOT da.daempty AND                                    041
        NOT goodrng(da.dacsp) THEN                               042
            BEGIN                                                043
            da.dacsp.stpsid _ origin;                            044
            da.daccnt _ 1;                                       045
            END;                                                 046
    %initialize frzarray pointer%                                047
        ptr _ 0;                                                 048
    %initialize first commands block to empty%                   049
        cmdinit($commands, &da);                                 050
    %..initialize marker flag to false..%                        051
        mkrflg _ FALSE;                                          052
    %..initialize vertical beam position..%                      053
        da.dacrow _ 0;                                           054
    IF NOT litreset THEN rstlit();                               055
    IF bmcnt THEN bmoff();                                       056
    clearda(&da);%clear the display image %                      057
    clrall(&da, TRUE); % zero LSRT entries %                     058
%..Display Generation..%                                         059
    %..initialize LSRT (line segment reference table) pointer..%
                                                                 060
        rttop _ &rt _ da.dalsrt; %start of LSRT%                 061
        rte _ da.dalsz * lsrtl + &rt; %end of LSRT%              062
        rtsize _ da.dalsz;                                       063
    IF da.daempty OR da.dacsp = endfil THEN                      064
        BEGIN                                                    065
        damt(&da);                                               066
        RETURN(endfil, 1, TRUE);                                 067
        END;                                                     068
    %set values to zero so system will blow up if they are used%
                                                                 069
        da.dalsrt _ da.dalsz _ 0;                                070
    frzfrmt(&da);                                                071
    &sa _  IF &oldsw THEN &oldsw                                 072
        ELSE openseq (fmtstd _ da.dacsp, seqend(fmtstd, sv1 _
        da.davspec, sv2 _ da.davspc2), sv1, sv2, da.dausqcod,
        da.dacacode);                                            073
    charct _ da.daccnt;                                          074
    UNTIL da.dacrow > da.damrow OR &rt >= rte                    075
    OR (stid _ seqgen (&sa)) = endfil DO                         076
        BEGIN                                                    077
```

```
        stfrmt (&da, stid, charct, sa.swclvl, sa.swslvl,
        sa.swsvw, sa.swvspec, sa.swvsp2);                          078
        charct _ 1;                                                079
        frzntab(stid, $frzarray, $ptr); %freeze textblock if stid
        is from a file and it hasn't been frozen yet%          08485
        IF nldevice = devlproc THEN %temporary patch for LP%   089
           BEGIN                                                   090
           prcmds(&da);                                            091
           cmdinit($commands, &da);                                092
           END;                                                    093
        END;                                                       094
     prcmds(&da); %process the accumulated display JSYS commands%
                                                                   095
     % If endflg is FALSE, we haven't reached the end of the
     material to be displayed  befoe the end of the screen. % 096
     endflg _                                                      097
        IF stid # endfil AND da.dacrow > da.damrow THEN FALSE 098
        ELSE TRUE;                                                 099
     da.davspec _ sa.swvspec;                                     0100
     da.davspc2 _ sa.swvsp2;                                      0101
     dspvsp(da.davspec, da.davspc2, 3);                           0102
     stid _ dgstid;  %dgstid, dgstml, dgstl set by stfrmt%        0103
     charct _ dgstl;                                              0104
  %..termination..%                                               0105
     UNTIL (ptr _ ptr - 1) < 0 DO                                 0106
        frzblk(frzarray[ptr], -1);                                0107
     %see if screen is really empty - could be i-viewspec didn't
     pass any statements%                                         0108
        da.dalsrt _ &rt; %starting address for damt to format
        into%                                                     0109
        IF da.dacrow = 0  OR                                      0110
        (da.dacrow = da.davinc AND                                0111
        [da.dalsrt].rtsrce = srcdot) THEN                         0112
           BEGIN                                                  0113
           damt(&da);                                             0114
           stid _ endfil;                                         0115
           charct _ 1;                                            0116
           END;                                                   0117
     IF NOT &oldsw THEN closeseq (&sa);                           0118
  %restore da LSRT table address, size %                          0119
     da.dalsrt _ rttop; %it may have changed%                     0120
     da.dalsz _ rtsize;                                           0121
  RETURN(stid, charct, endflg);                                   0122
  END.                                                            0123
                                                                  0124
(frzntab)PROCEDURE (stid, array, atop);                        08481
  %called by dafrmt and daupdate to freeze textblock, but only if
  it isn't already in "array"%                                 08482
  LOCAL fpgindx, frzindx;                                      08486
  REF array, atop;                                             08487
  IF NOT stid.stastr THEN                                        080
     BEGIN %freeze the page in core if not already frozen%       081
     IF NOT (fpgindx _ lodprop( stid, txttyp)) THEN              082
        err($"No text block associated with node");            06293
     %check if this page already frozen%                         083
     FOR frzindx _ 0 UP 1 UNTIL >= atop DO                       084
```

```
        IF array[frzindx] = fpgindx THEN EXIT;                          085
      IF frzindx >= atop THEN %not previously frozen by dafrmt%
                                                                        086
        frzblk(array[atop := atop + 1] _ fpgindx, 1);                   087
      END;                                                              088
    RETURN;                                                          08486
    END.                                                             08484
(daupdate) PROCEDURE (da); %update display area image%               08591
    %This routine updates the display image in the given text
    display area (da) by reformating those statements which have
    changed.%                                                        08592
    %Wherever possible, statements on the current screen are copied
    over to the new screen.  Globals cdtype, cdstd1, cdstd2 and
    cdstop describe the kind of change that has taken place in the
    file.%                                                           08593
    %the reason for all the messing around with table pointers is
    that if stfrmt gets called, the new table being fomatted can
    change physical locations and size (more than once)
    NOTE:  assumptions are made that imply that only statements on
    the screen have been edited.%                                    08594
    %----------------%                                              08595
    LOCAL                                                            08596
      deleted,    %flag used to see what's missing from new table%
                                                                     08597
      ls,         %current line segment entry%                       08598
      ols,        %old line segment entry%                           08599
      nls,        %next line segment entry%                          08600
      origls,     %original line segment entry%                      08601
      sa,         %sequence work area%                               08602
      stid,       %current STID being considered for formatting%
                                                                     08603
      bp,   %byte pointer%                                           08604
      length,                                                        08605
      endstd,        % End stid in sequence to be formatted. %
                                                                     08606
      svstd,                                                         08607
      sv1,    %Viewspec word 1 saved from first sequence work
      area. %                                                        08608
      sv2,    %Viewspec word 2 saved from first sequence work
      area. %                                                        08609
      slv1,        %base level saved from first wequence work
      area%                                                          08610
      match,      %flag: whether srch found the STID in old table%
                                                                     08611
      litup,      %flag: TRUE if lit was being displayed%            08612
      nextsup,    %addr of next ls to suppress for literal
      feedback:  for line processor%                                 08613
      copyflag,   %TRUE if lsrt entries copied from old to new
      lsrt%                                                          08614
      newtop,     %new logical top to minimize searching old
      table%                                                         08615
      tabltop,    %start of search in old table%                     08616
      newaulsrt,  %temp for new aux table allocation%                08617
      rtsave,     %da.dalsrt on entry%                               08618
      ptr,        %current (+1) end of frzarray%                     08619
      frzarray[40]; %list of frozen pages formatted but not
```

```
    processed (prcmds)%                                          08620
REF nls, da, ls, ols, origls, sa, rt, art, mkrptr;               08621
                                                                 08622
IF NOT da.daexis THEN err($"Fatal display error in DAUPDATE");
                                                                 08623
IF bmcnt THEN bmoff();                                           08624
IF NOT da.davspec.vsdaft THEN RETURN; %defer recreate display%
                                                                 08625
   %                                                             08626
   this block former defer recreate code; caused loop when
   inserting more than one line of text with VS v on:            08627
   BEGIN                                                         08628
   IF lplitreset THEN                                            08629
      BEGIN                                                      08630
      litline _ lplitline;                                       08631
      litreset _ FALSE;                                          08632
      litapflag _ TRUE;                                          08633
      rstlit();                                                  08634
      lplitreset _ FALSE;                                        08635
      END;                                                       08636
   IF NOT litreset THEN rstlit();                                08637
   RETURN;                                                       08638
   END;                                                          08639
   %                                                             08640
%..INITIALIZATION....%                                           08641
   %..if csp is not valid, set to origin..%                      08642
      IF NOT da.daempty AND                                      08643
      NOT goodrng(da.dacsp) THEN                                 08644
         BEGIN                                                   08645
         da.dacsp.stpsid _ orgstid.stpsid;                       08646
         da.daccnt _ 1;                                          08647
         dafrmt(&da, 0);                                         08648
         RETURN;                                                 08649
         END;                                                    08650
   %..terminal device check..%                                   08651
      IF nlmode = typewriter THEN err($"display attempted on
      typewriter terminal
      - proceed at own risk");                                   08652
      IF NOT scwindow AND nldevice = devlproc AND cdtype =
      dspjpf AND (da.dalneighbor OR da.darneighbor OR
      da.datneighbor OR da.dabneighbor) THEN                     08653
         RETURN (dafrmt (&da, 0));                               08654
   %..update previous file in da, for jump routines%             08655
      da.dapstf _ da.dacsp.stfile;                               08656
   %..initialize marker flag to false..%                         08657
      mkrflg _ FALSE;                                            08658
   %initialize frzarray pointer%                                 08659
      ptr _ 0;                                                   08660
   %initialize first commands block to empty%                    08661
      cmdinit($commands, &da);                                   08662
   %make sure aux big enough%                                    08663
      IF aulsize < da.dalsz THEN                                 08664
         BEGIN                                                   08665
         IF NOT (newaulsrt _ getblk(da.dalsz * lsrtl, $dspblk))
         THEN %we will format into a table smaller than the old
         one - not intention of design%                          08666
```

```
            dismes(1,$"Possibly fatal DISPLAY ERROR...
            Enter a viewspec 'f' and check your screen.") 08667
         ELSE                                                08668
            BEGIN                                            08669
            freeblk(aulsrt - bhl, $dspblk); %deallocate old aux
            table%                                           08670
            aulsrt _ newaulsrt + bhl;%update global for aux
            LSRT%                                            08671
            END;                                             08672
         END;                                                08673
      %fix up lit area%                                      08674
         IF nldevice = devlproc AND da.dalssup THEN          08675
            BEGIN                                            08676
            litup _ TRUE;                                    08677
            nextsup _ da.dalsrt + (da.dalssup := 0)*lsrtl;   08678
            END                                              08679
         ELSE litup _ FALSE;                                 08680
         IF NOT litreset THEN rstlit();                      08681
         lplitreset _ FALSE;                                 08682
      %do empty screen and return%                           08683
         IF da.daempty OR da.dacsp = endfil THEN             08684
            BEGIN                                            08685
            damt(&da);                                       08686
            RETURN;                                          08687
            END;                                             08688
      copyflag _ FALSE;                                      08689
      %..initialize LSRT (line segment reference table) pointers
      to use auxilary LSRT..%                                08690
         oldlsrt _ rtsave _ tabltop _ newtop _ &art  _ da.dalsrt;
                                                             08691
         arte _ da.dalsz * lsrtl + &art; %end + 1%           08692
         rttop _ &rt _ aulsrt;                               08693
         rte _ (rtsize _ aulsize) * lsrtl + aulsrt; %end + 1%
                                                             08694
      %initialize LP links to 0%                             08695
         FOR &ls _ &art UP lsrtl UNTIL >= arte DO            08696
            ls.rtlplink _ 0;                                 08697
      %..initialize vertical beam position..%                08698
         da.dacrow _ 0;                                      08699
      %..clear auxillary lsrt..%                             08700
         DO rt _ 0 UNTIL (&rt _ &rt + 1) = rte;              08701
         &rt _ aulsrt;                                       08702
   %..image update code..%                                   08703
      %..take care of frozen statements first...%            08704
      IF da.davspec.vsfrzf THEN                              08705
         BEGIN                                               08706
         frzfrmt(&da);                                       08707
         %set effective top of LSRT past frozen statements%  08708
         DO IF art.rtsrce = srcdot THEN tabltop _ &art + lsrtl
                                                             08709
         UNTIL NOT art.rtexis OR (&art _ &art + lsrtl) >= arte;
                                                             08710
         END;                                                08711
      %..start sequence at top of screen ..%                 08712
         &sa _ openseq(svstd _ da.dacsp, endstd _ seqend( svstd,
         sv1 _ da.davspec, sv2 _ da.davspc2 ), sv1, sv2,
```

```
      da.dausgcod, da.dacacode);                                    08713
      match _ FALSE; %no match between sequence & LSRT yet%
                                                                    08714
   %...MAIN FORMATTING LOOP....%                                    08715
      UNTIL da.dacrow > da.damrow OR (stid _ seggen (&sa)) =
      endfil OR &rt >= rte                                          08716
      DO                                                            08717
         BEGIN                                                      08718
         IF stid = cdstd1 OR stid = cdstd2 THEN %must be
         reformatted%                                               08719
            BEGIN                                                   08720
            &ls _ &origls _ &rt;                                    08721
            stfrmt (&da, stid, 1, sa.swclvl, sa.swslvl,
            sa.swsvw, sa.swvspec, sa.swvsp2);                       08722
            frzntab(stid, $frzarray, $ptr); %freeze text block%
                                                                    08723
            IF da.davspec.vsblkf AND ls.rty < art.rty THEN
            cline (da.daleft, da.dacrow+da.datop-da.davinc,
            da.daright-da.daleft);                                  08724
            IF srch (stid, sa.swclvl, match, tabltop) THEN
                                                                    08725
               BEGIN % This statement was on the screen before,
               see if we can eliminate some of the line
               segments%                                            08726
               &ols _ &art;                                         08727
               FOR &ls UP lsrtl UNTIL >= &rt DO                     08728
                  BEGIN  %compare corresponding line segements,
                  if they are the same and at same location,
                  dont rewrite them%                                08729
                  bp _ chbptr(ols.rtccnt - 1) + ( IF stid =
                  cdstd1 THEN $cdstr1 ELSE $cdstr2 );               08730
                  IF ls.rtx1 = ols.rtx1
                  AND ls.rty = ols.rty
                  AND ls.rtx2 = ols.rtx2
                  AND (length _ slngth(ls.rtbps, ls.rtbpe)) =
                  slngth(ols.rtbps, ols.rtbpe)
                  AND complsg(ls.rtbps, bp, length)
                  AND NOT (litup AND &ols < nextsup)
                  %suppressed%
                  THEN %dont really need to write it!!%             08731
                     BEGIN                                          08732
                     ls.rtlsid _ ols.rtlsid;                        08733
                     ls.rtnew _ FALSE;                              08734
                     copyflag _ TRUE;                               08735
                     END                                            08736
                  ELSE %must really write this one, so make
                  sure its neighbors on the same line also get
                  written for LP's%                                 08737
                     IF nldevice = devlproc THEN                    08738
                        BEGIN                                       08739
                        FOR &nls _ &ls - lsrtl DOWN lsrtl UNTIL
                        < &origls DO                                08740
                           IF nls.rty = ls.rty THEN                 08741
                              BEGIN                                 08742
                              IF NOT nls.rtnew THEN                 08743
                                 BEGIN                              08744
```

```
                                nls.rtnew _ TRUE;                   08745
                                nls.rtlsid _ 0;                     08746
                                END;                                08747
                             END                                    08748
                          ELSE EXIT LOOP;                           08749
                       FOR &nls _ &ls + lsrtl UP lsrtl UNTIL
                       >= &rt DO                                    08750
                          IF nls.rty = ls.rty THEN                  08751
                             BEGIN                                  08752
                             &ols _ &ols + lsrtl;                   08753
                             &ls _ &nls;                            08754
                             IF &ols >= arte THEN EXIT LOOP;
                                                                    08755
                             END                                    08756
                          ELSE EXIT LOOP;                           08757
                       END;                                         08758
                    &ols _ &ols + lsrtl;                            08759
                    IF &ols >= arte THEN EXIT LOOP;                 08760
                    END;                                            08761
                 END;                                               08762
              END                                                   08763
           ELSE                                                     08764
              BEGIN %search for match in old LSRT%                  08765
              IF (match _ srch (stid, sa.swclvl, match, tabltop :
              newtop)) THEN                                         08766
                 BEGIN                                              08767
                 IF newtop AND stid NOT= cdstop THEN tabltop _
                 newtop;                                            08768
                 IF (litup AND &art < nextsup) OR NOT
                 (lscopy(TRUE,&da)) THEN                            08769
                    BEGIN                                           08770
                    stfrmt (&da, stid, 1, sa.swclvl, sa.swslvl,
                    sa.swsvw, sa.swvspec, sa.swvsp2);               08771
                       %reformat if partial stmt found or
                       restoring supressed statements%              08772
                    frzntab(stid, $frzarray, $ptr); %freeze text
                    block%                                          08773
                    END                                             08774
                 ELSE copyflag _ TRUE;                              08775
                 END                                                08776
              ELSE %no match%                                       08777
                 BEGIN                                              08778
                 stfrmt (&da, stid, 1, sa.swclvl, sa.swslvl,
                 sa.swsvw, sa.swvspec, sa.swvsp2);                  08779
                 frzntab(stid, $frzarray, $ptr);  %freeze text
                 block%                                             08780
                 END;                                               08781
              END;                                                  08782
           IF stid = cdstop AND cdstop # endfil AND NOT (litup
           AND lstidnxt(&art) < nextsup %more suppressed stmts% )
           THEN                                                     08783
              %no need to reformat (stfrmt) rest of LSRT%           08784
              BEGIN                                                 08785
              %if old table has useful information, copy all of
              it%                                                   08786
                 IF srch(stid, sa.swclvl, match, tabltop) THEN
```

```
                                                                    08787
                    BEGIN                                           08788
                    &art _ lstidnxt (&art);                         08789
                       %bump lsrt source addr%                      08790
                    match _ TRUE;                                   08791
                    copyflag _ TRUE;                                08792
                    lscopy(FALSE,&da); %copy rest of LSRT%          08793
                    END                                             08794
                 ELSE match _ FALSE; %flag for no copy done%
                                                                    08795
              IF da.dacrow <= da.damrow AND &rt < rte THEN %more
              room on screen%                                       08796
                 BEGIN %get and reformat new statements%            08797
                 IF match THEN %copy caused sequence to be
                 broken%                                            08798
                    BEGIN                                           08799
                    % save viewspecs from sequence work area in
                    case user seq code canged them. %               08800
                       sv1 _ sa.swvspec;                            08801
                       sv2 _ sa.swvsp2;                             08802
                    %save base level from sequence work area in
                    case changed by impending openseq%              08803
                       slv1 _ sa.swslvl;                            08804
                    %close last sequence%                           08805
                       closeseq(&sa);                               08806
                    %open new sequence at last stid copied%         08807
                       &sa _ openseq ([&rt - lsrtll.rtstid,
                       endstd,  sv1, sv2, da.dausqcod,
                       da.dacacode);                                08808
                       stid _ seqgen(&sa);%already
                       formatted-throw away%                        08809
                    %restore base level for this screen%            08810
                       sa.swslvl _ slvl;                            08811
                    END;                                            08812
                 %do regular format on stmts now able to fit on
                 screen for first time%                             08813
                    UNTIL &rt >= rte OR da.dacrow > da.damrow OR
                    (stid _ seqgen (&sa)) = endfil DO               08814
                       stfrmt (&da, stid, 1, sa.swclvl,
                       sa.swslvl, sa.swsvw, sa.swvspec,
                       sa.swvsp2);                                  08815
                 END;                                               08816
              EXIT;                                                 08817
              END; %case where old screen can be used%              08818
           END; %..OF MAIN LOOP....%
                                                                    08819
        %close whatever sequence it is on%                          08820
           da.davspec _ sa.swvspec;                                 08821
           da.davspc2 _ sa.swvsp2;                                  08822
           dspvsp(da.davspec, da.davspc2, 3);                       08823
           closeseq(&sa);                                           08824
        %Construct movelsg and deletlsg commands and update LSRT%
                                                                    08825
           IF copyflag THEN                                         08826
              BEGIN                                                 08827
              %initialize pointers%                                 08828
```

```
            &art _ rttop;                                          08829
            arte _ rtsize * lsrtl + rttop;                         08830
            rte _ da.dalsz  * lsrtl + rtsave;                      08831
         UNTIL NOT art.rtexis OR                                   08832
         &art >= arte DO                                           08833
            BEGIN                                                  08834
            IF NOT art.rtnew THEN                                  08835
               BEGIN                                               08836
               &rt _ rtsave; %old table%                           08837
               DO                                                  08838
                  IF art.rtlsid = rt.rtlsid AND                    08839
                     art.rty NOT= rt.rty THEN                      08840
                     BEGIN                                         08841
                     pushdc(movelsg, &art);                        08842
                     IF nldevice NOT= devlproc THEN                08843
                        cklsfreestring(&art);                      08844
                           %moving causes the first byteptr to
                           be set -1, so we must free th string
                           now (while we still know where it
                           is)%                                    08845
                     EXIT;                                         08846
                     END                                           08847
               UNTIL (&rt _ &rt + lsrtl) >= rte OR                 08848
               NOT rt.rtexis;                                      08849
               END;                                                08850
            &art _ &art + lsrtl;                                   08851
            END;                                                   08852
         %construct deletlsg commands%                             08853
            &rt _ rtsave; %old table%                              08854
            DO                                                     08855
               BEGIN                                               08856
               &art _ rttop;                                       08857
               deleted _ TRUE;                                     08858
               DO                                                  08859
                  IF rt.rtlsid = art.rtlsid THEN                   08860
                     BEGIN                                         08861
                     deleted _ FALSE;                              08862
                     EXIT;                                         08863
                     END                                           08864
               UNTIL (&art _ &art + lsrtl) >= arte OR
               NOT art.rtexis;                                     08865
               IF deleted THEN                                     08866
                  BEGIN                                            08867
                  cklsfreestring(&rt); %see if it is allocated
                  string to deallocate%                            08868
                  pushdc(deletlsg, &rt);                           08869
                  END;                                             08870
               END                                                 08871
            UNTIL (&rt _ &rt + lsrtl) >= rte OR
            NOT rt.rtexis;                                         08872
         END                                                       08873
      ELSE %no lsrt entries were copied, thus just clear out
      old image and free any strings%                              08874
         BEGIN                                                     08875
         pushdc(blankda, 0);                                       08876
         &art _ rtsave; %arte is still set up%                     08877
```

```
            DO cklsfreestring(&art)                                    08878
               UNTIL (&art _ &art + lsrtl) >= arte OR NOT
               art.rtexis;                                             08879
            END;                                                       08880
         %process any commands that were stacked%                      08881
            prcmds(&da);                                               08882
         %release frozen pages%                                        08883
            UNTIL (ptr _ ptr - 1) < 0 DO                               08884
               frzblk(frzarray[ptr], -1);                              08885
         %..clear rest of current LSRT...%                             08886
            BUMP DOWN &rt;                                             08887
            UNTIL (&rt _ &rt + 1) >= rte DO rt _ 0;                    08888
      %..termination..%                                                08889
         IF da.dacrow = 0  OR                                          08890
         (da.dacrow = da.davinc AND                                    08891
         rttop.rtsrce = srcdot) THEN                                   08892
            damt(&da);                                                 08893
      %transpose table pointers so aux becomes real LSRT%              08894
         aulsrt _ rtsave; %old da.dalsrt%                              08895
         da.dalsrt _ rttop; %current (new) table%                      08896
         aulsize _ da.dalsz; %old size%                                08897
         da.dalsz _ rtsize; %new size%                                 08898
      RETURN;                                                          08899
      END.                                                             08900
                                                                       08901
   (cklsfreestring)PROCEDURE(ls); %if line seg is from string
   allocated from dspblk then deallocate it%                            0378
      REF ls;                                                          0379
      CASE ls.rtsrce OF                                                0380
         =srcstno, =srcsig:                                            0381
            freestring (ls.rtbps.bpadr -1, $dspblk); %freestring
            checks bounds for calls involving dspblk%                  0382
         ENDCASE NULL;                                                 0383
      RETURN; END.                                                     0384
                                                                       0385
   (damt)     PROCEDURE (da); %empty da routine%                       0386
      %Thise routine constructs the empty message in a text            0387
      display area%                                                    0388
      %------------%                                                   0389
      LOCAL ls, end;                                                   0390
      REF ls, da;                                                      0391
      IF da.dacrow > 0 THEN                                            0392
         BEGIN                                                         0393
         clearda(&da); %clear display area %                           0394
         clrall(&da, TRUE);%zap lsrt entries%                          0395
         END;                                                          0396
      &rt _ da.dalsrt;                                                 0397
      end _ &rt + lsrtl;                                               0398
      DO rt _ 0 UNTIL (&rt _ &rt + 1) = end;                           0399
      &rt _ &rt - lsrtl;                                               0400
      rt.rtexis _ rt.rtnew _ rt.rtllsg _ TRUE;                         0401
      rt.rtsrce _ srcast;                                              0402
      rt.rtfont _ fnormal;                                             0403
      rt.rtstid.stastr _ TRUE;                                         0404
      rt.rtstid.stadr _ $mt;                                           0405
      rt.rtccnt _ 1;                                                   0406
```

```
    rt.rthinc _ da.dahinc;                                          0407
    rt.rtcsize _ da.dacsize;                                        0408
    rt.rtx1 _ da.dahinc;                                            0409
    rt.rtx2 _ gapcol _  MAX(rt.rtx1, MIN(rt.rtx1 +
    (mt.L-1)*da.dahinc, da.daright-da.daleft));                     0410
    rt.rty _ da.dacrow _ 0;                                         0411
    rt.rtbps _ chbmty + $mt;                                        0412
    rt.rtbpe _ gapbp _ chbptr(mt.L) + $mt;                          0413
    wrtstr(&da, &rt);                                               0414
    RETURN;                                                         0415
    END.                                                            0416
                                                                    0417
(frzfrmt)  PROCEDURE (da);  %display frozen statements%             0418
    %This routine formats the frozen statements for a               0419
    display area%                                                   0420
    %------------------%                                           0421
    LOCAL  ls, fz, davs1, davs2, sa, stid;                          0422
    REF  ls, da, fz, sa;                                            0423
    %display the frozen statements if viewspec is                   0424
    on--otherwise return%                                           0425
       IF NOT da.davspec.vsfrzf THEN RETURN;                        0426
    davs1 _ da.davspec; %save view specs from display area record%
                                                                    0427
    davs2 _ da.davspc2;                                             0428
    IF &fz _ da.dafrzl THEN %there are frozen statements%           0429
       DO                                                           0430
          IF fz.fzexis THEN                                         0431
             BEGIN  %statement frozen...display it%                 0432
             % set up a sequence that will only return the first
             statement if it passes -- by setting the L viewspec to
             E + 0 %                                                0433
                da.davspec _ <PRMSPC, setlt>('e, fz.fzvspec,
                fz.fzvspc2 : da.davspc2);                           0434
             &sa _ openseq (fz.fzstid, fz.fzstid, da.davspec,
             da.davspc2, da.dausqcod, da.dacacode);                 0435
             UNTIL (stid _ seqgen (&sa)) = endfil DO                0436
                stfrmt(&da, stid, 1, sa.swclvl, sa.swslvl,
                sa.swsvw, sa.swvspec, sa.swvsp2);                   0437
             closeseq (&sa);                                        0438
             IF da.dacrow > da.damrow THEN RETURN; %filled display
             area%                                                  0439
             END                                                    0440
          ELSE err($"NLS error: Frozen statement list incorrect
          Turn off viewspec o")                                     0441
       UNTIL (&fz _ fz.fznext) = 0;                                 0442
    da.davspec _ davs1; %restore original view specs%               0443
    da.davspc2 _ davs2;                                             0444
    %display dotted line%                                           0445
    rt.rtbps _ chbmty + $dotstr;                                    0446
    rt.rtbpe _ gapbp _ chbptr(dotstr.L) + $dotstr;                  0447
    rt.rty _ da.dacrow := da.dacrow + da.davinc;                    0448
    rt.rtx1 _ 0;                                                    0449
    rt.rtx2 _ gapcol _  MAX(rt.rtx1, MIN(rt.rtx1 +
    (dotstr.L-1)*da.dahinc, da.daright-da.daleft));                 0450
    rt.rt1lsg _ rt.rtexis _ TRUE;                                   0451
    rt.rtcsize _ da.dacsize;                                        0452
```

```
      rt.rthinc _ da.dahinc;                                          0453
      rt.rtsrce _ srcdot;                                             0454
      pushdc(writelsg, &rt);                                          0455
      nxtlsg(&da);                                                    0456
      RETURN;                                                         0457
      END.                                                            0458
                                                                      0459
%....Display format support....%                                      0460
   (stfrmt)  PROC (da, stid, charct, level, origlev,  stvec,
   viewspec1, vwspc2);  %statement format%                            07660
      %This routine formats a statement according to the information
      in the text display area record, da, the level of the
      statement, level, and the statement vector work area, stvec.
      The formating begins at the character indicated by charct.%
                                                                      07661
      %NOTE:  special handling of indentation--if indent=OFF and
      branch/plex only, integrity of structure maintained%           07662
      %viewspec1 is first word of viewspecs, as obtained from seggen%
                                                                      07663
      %The global 'rt' is the current line segment referencetable
      entry%                                                          07664
      %--------------%
                                                                      07665
      LOCAL                                                           07666
         bptr,            %temp for saving byte pointer values%
         stringaddr,       %temp for string address values%          07667
         numberadr,       %temp for stmt # address%                  07668
         char,            %return from lsgfrmt%                      07669
         numflg,                                                      07670
         sdbadr,                                                      07671
         word,                                                        07672
         trunc,             %current line for truncation viewspecxx%
                                                                      07673
         tabcol,                                                      07674
         indent,            %left margin offset%                      07675
         ls,                %line segment address%                    07676
         header,                                                      07677
         length,                                                      07678
         right,                                                       07679
         left,                                                        07680
         blnklnf,    %flag to note blank line already out%           07681
         wa[10];                                                      07682
      REF da, sdbadr, ls, stringaddr, numberadr;
                                                                      07683
      ON SIGNAL                                                       07684
         =spacerr: %could not allocate such a large block%           07685
            BEGIN                                                     07686
            rt.rtexis _ FALSE; %make last one not exist in case it is
            incomplete%                                               07687
            GOTO stfrend; %entry at &rt has not been pushed on
            commands list yet%                                        07688
            END;                                                      07689
      ELSE NULL;
                                                                      07690
      udpnwrap _ 0;                                                   07691
      IF NOT da.daexis THEN                                           07692
```

```
      err($"Fatal display error in STFRMT");                          07693
   IF stid = endfil THEN RETURN;                                      07694
   dgstid _ stid;                                                     07695
   IF NOT stid.stastr THEN                                            07696
      BEGIN                                                           07697
      lodprop(stid, txttyp : &sdbadr);                                07698
      dgstml _ sdbadr.schars;                                         07699
      END                                                             07700
   ELSE dgstml _ [stid.stadr].L;                                      07701
   lastch _ ENDCHR;                                                   07702
      %for special case of null line%                                 07703
   indent _        %indentation%                                      07704
      CASE TRUE OF                                                    07705
         = (viewspec1.vsrind AND ( viewspec1.vsbrof OR
         viewspec1.vsplxf)):                                          07706
            MAX (0, MIN (da.daind * (level-origlev), da.damind));
                                                                      07707
         # viewspec1.vsindf: 0;                                       07708
         ENDCASE                                                      07709
            MAX (0, MIN (da.daind * (level-1), da.damind));           07710
   IF charct <= 1 THEN                                                07711
      BEGIN                                                           07712
      charct _ 1;                                                     07713
      IF NOT viewspec1.vsnamf THEN %do not show statement name%
                                                                      07714
         IF stid.stastr THEN %a-string%                               07715
            BEGIN                                                     07716
            IF NOT da.dacsp.stastr THEN                               07717
               BEGIN %try to skip over text that looks like a
               name%                                                  07718
               %this is bullshit code--probably should be removed
               or replaced%                                           07719
               wa _ stid;                                             07720
               wa[1] _ 1;                                             07721
               fechc1(forward, $wa);                                  07722
               header _ filhdr(da.dacsp.stfile);                      07723
                  %wont work for frozen statement a-strings           07724
                  from another file%                                  07725
               left _ [$namdl1-$filhed+header];                       07726
               right _ [$namdl2-$filhed+header];                      07727
               xtrnam($stn, $wa, left, right);                        07728
               IF stn.L = empty THEN charct _ 1 %no name found%
                                                                      07729
               ELSE charct _ wa[1];                                   07730
               END;                                                   07731
            END                                                       07732
         ELSE charct _ sdbadr.sname;%skip over name%                  07733
      END;                                                            07734
   dgstl _ charct - 1;                                                07735
   %initialize LSRT entry%                                            07736
      rt.rtx1 _ 0;                                                    07737
      rt.rty _ da.dacrow;                                             07738
      rt.rtexis _ rt.rtnew _ TRUE;                                    07739
      rt.rt1lsg _ FALSE;                                              07740
      rt.rtlsid _ 0;                                                  07741
      rt.rthinc _ da.dahinc;                                          07742
```

```
        rt.rtcsize _ da.dacsize;                                        07743
        rt.rtstid _ stid;                                               07744
        rt.rtccnt _ charct;                                             07745
        rt.rtcbug _ 1;                                                  07746
        rt.rtsrce _ srcnul;                                             07747
        rt.rtfont _ fnormal;                                            07748
        rt.rtlev _ level;                                               07749
        rt.rtx2 _ da.damcol / rt.rthinc * rt.rthinc; %round down
        damcol for stop coord%                                          07750
        rt.rtpartst _ TRUE;%assume this segment not last in
        statement%                                                      07751
        %set up byte pointers for reading characters from string or
        statement%                                                      07752
        IF NOT stid.stastr THEN                                         07753
           rt.rtbps _ rt.rtbpe _ stbptr(charct - 1) + &sdbadr +
           sdbhdl                                                       07754
        ELSE                                                            07755
           rt.rtbps _ rt.rtbpe _ chbptr(charct - 1) + stid.stadr;
                                                                        07756
     IF NOT stid.stastr AND viewspec1.vsstnf AND
     (stid.stpsid NOT= origin OR viewspec1.vssidf) THEN                 07757
        BEGIN                                                           07758
        &numberadr _ getstring(30, $dspblk);                            07759
        IF viewspec1.vssidf  % display line #'s as sids %               07760
        THEN                                                            07761
           *numberadr* _ '0, STRING( getsid(stid) )                     07762
        ELSE                                                            07763
           fechnm(stvec, &numberadr);                                   07764
              %convert statement vector to string%                      07765
        IF NOT viewspec1.vsstnr THEN                                    07766
        BEGIN %statement number on left%                                07767
        numflg _ FALSE;                                                 07768
        rt.rtbps _ chbmty + &numberadr;                                 07769
        rt.rtbpe _ chbptr(numberadr.L) + &numberadr;                    07770
        rt.rtsrce _ srcstno;                                            07771
        rt.rthinc _ da.danohi;                                          07772
        rt.rtcsize _ da.danocs;                                         07773
        gapcc _ 0;                                                      07774
        gapcol _ da.daccol;                                             07775
        gapbp _ rt.rtbpe;                                               07776
        char _  dgstl; %save current char count%                        07777
        &ls _ nxtlsg(&da);                                              07778
        dgstl _  char; %restore current char count%                     07779
        ls.rtx1 _ indent;                                               07780
        ls.rtx2 _ MIN(ls.rtx1 + (numberadr.L-1) * ls.rthinc,
        da.damcol - ls.rthinc); %for bug selection%                     07781
        IF da.damcol - ls.rtx1 < numberadr.L*ls.rthinc THEN             07782
           (setxcm):                                                    07783
           *numberadr*[(da.damcol - ls.rtx1)/ls.rthinc] _ '!;
                                                                        07784
           %to note that some of stmt # is lost%                        07785
        ls.rt1lsg _ FALSE;                                              07786
        rt.rt1lsg _ FALSE;                                              07787
        IF ls.rtx2 + 2*da.dahinc < da.damcol THEN                       07788
           da.daccol _ rt.rtx1 _
                ls.rtx2 + 2*da.dahinc - indent                          07789
```

```
            ELSE                                                    07790
               BEGIN                                                07791
               ls.rtilsg _ TRUE;                                    07792
               rt.rty _ da.dacrow := da.dacrow + da.davinc;         07793
               rt.rtx1 _ da.daccol _ 0;                             07794
               END;                                                 07795
            pushdc(writelsg, &ls);                                  07796
            rt.rtcsize _ da.dacsize;                                07797
            rt.rthinc _ da.dahinc;                                  07798
            rt.rtbps _ rt.rtbpe _ stbptr(charct - 1) + &sdbadr +
            sdbhdl;                                                 07799
            rt.rtccnt_charct;                                       07800
            END                                                     07801
         ELSE %numbers on right%                                    07802
            numflg _ TRUE;                                          07803
         END                                                        07804
      ELSE numflg _ FALSE; %no statement number%                    07805
      blnklnf _ FALSE; %init; TRUE if stmt # at right on separate
      line%                                                         07806
      IF vwspc2.vsmkrf AND NOT stid.stastr THEN                     07807
         BEGIN %show markers%                                       07808
         &mkrptr _ $mkrtb - $filhed +                               07809
            (header _ filhdr(stid.stfile));                         07810
            % initialize to beginning of marker table %             07811
         mkrend _ mkrptr + ($mkrtbl-$filhed+header)*mkrl;           07812
            % current end of table %                                07813
         % initialize flag to false--do not look for markers       07814
         in this statement %                                        07815
            mkrflg _ FALSE;                                         07816
         DO %look for this statement's psid in the marker           07817
         table%                                                     07818
            IF mkrptr.mkpsid = stid.stpsid THEN                     07819
               BEGIN   % found one %                                07820
               % set 'look-for-markers' flag for this              07821
               statement%                                           07822
                  mkrflg _ TRUE;                                    07823
               %save chr count + 1%                                 07824
                  mkrcnt _ mkrptr.mkccnt + 1;                       07825
               % terminate loop %                                  07826
                  EXIT LOOP;                                        07827
               END                                                  07828
         UNTIL (mkrptr _ mkrptr + mkrl) > mkrend;                   07829
         END                                                        07830
      ELSE mkrflg _ FALSE;                                          07831
      %initialize truncation counter%                               07832
         trunc _ viewspec1.vstrnc;                                  07833
      UNTIL (trunc := trunc - 1) <= 0 OR
      &rt >= rte OR
      da.dacrow > da.damrow DO %format lines for the statement%     07834
         (stfrbegin):                                               07835
         BEGIN %once per line%                                      07836
         rt.rtx1 _ da.daccol _ indent + rt.rtx1;                    07837
         rt.rty _ da.dacrow := da.dacrow + da.davinc;               07838
         %format a line%                                            07839
            UNTIL &rt >= rte DO                                     07840
               BEGIN                                                07841
```

```
            rt.rtsrce _ srcstat;                                    07842
            CASE (char _ lsgfrmt(&da)) OF                           07843
               = ENDCHR: %end of input%                             07844
                  BEGIN  %finish up and return%                     07845
                  IF (NOT numflg OR (numflg AND rt.rtx2 # rt.rtx1
                  AND da.daccol >= rt.rtx2 -
                  (numberadr.L)*rt.rthinc)) THEN rt.rt1lsg _ TRUE;
                                                                    07846
                     %if statement number to be processed, this is
                     not last segment in this line, otherwise it
                     is%                                            07847
                  rt.rtexis _ TRUE;                                 07848
                  &ls _ nxtlsg(&da);                                07849
                  IF rt.rt1lsg AND NOT (numflg OR viewspec1.vsidtf
                  OR viewspec1.vsblkf) THEN ls.rtpartst _ FALSE;
                                                                    07850
                  %statement complete %                             07851
                  pushdc(writelsg, &ls);                            07852
                  trunc _ 0;                                        07853
                  EXIT LOOP;                                        07854
                  END;                                              07855
               = SP, = CR: %end of line%                            07856
                  BEGIN  %finish up and do next line%               07857
                  rt.rtexis _ TRUE;                                 07858
                  IF trunc > 0 OR (NOT numflg OR (numflg AND
                  da.daccol >= rt.rtx2 - (numberadr.L)*rt.rthinc))
                  THEN rt.rt1lsg _ TRUE;                            07859
                  &ls _ nxtlsg(&da);                                07860
                  IF (trunc <= 0 AND rt.rt1lsg) AND NOT (numflg OR
                  viewspec1.vsidtf OR viewspec1.vsblkf) THEN
                  ls.rtpartst _ FALSE; %statement now complete%
                                                                    07861
                  pushdc(writelsg, &ls);                            07862
                  rt.rtx1 _ 0;                                      07863
                  EXIT LOOP;                                        07864
                  END;                                              07865
               = TAB: %a TAB was input%                             07866
                  BEGIN  %continue if not end of line%              07867
                  %save global values across nxtlsg call - kludge%
                                                                    07868
                  char _ gapcc;                                     07869
                  bptr _ gapcol;                                    07870
                  %stfrmt starts daccol off at 0, while fndtab
                  uses da.dahinc for first position.  param to
                  fndtab and return from fndtab have to be
                  converted by one horizontal inc%                  07871
                  tabcol _ fndtab(&da, da.daccol/rt.rthinc +
                  udpwrapcol*udpnwrap) * rt.rthinc -
                  udpwrapcol*udpnwrap;                              07872
                  IF (tabcol > rt.rtx2 OR (udpwrapcol > 0 AND
                  tabcol > udpwrapcol)) THEN                        07873
                     BEGIN %too many columns--start new             07874
                     line%                                          07875
                     cdbckc();                                      07876
                     da.daccol _ da.daccol - rt.rthinc;
                     %apparently to satisfy lsgfrmt%                07877
```

```
                rt.rt1lsg _ TRUE; %last segment, this line%
                                                        07878
                &ls _ nxtlsg(&da);                      07879
                gapcc _ char;                           07880
                gapcol _ bptr;                          07881
                ls.rtexis _ TRUE;                       07882
                pushdc(writelsg, &ls);                  07883
                IF tabcol > rt.rtx2 THEN rt.rtx1 _ 0 %start
                next tab at left + indent%              07884
                ELSE                                    07885
                   BEGIN %wrap in middle of tab%        07886
                   rt.rtx1 _ tabcol - udpwrapcol - rt.rthinc;
                                                        07887
                   udpnwrap _ udpnwrap + 1;             07888
                   END;                                 07889
                EXIT LOOP;                              07890
                END;                                    07891
             &ls _ nxtlsg(&da);                         07892
             gapcc _ char;                              07893
             gapcol _ bptr;                             07894
             ls.rtexis _ TRUE;                          07895
             pushdc(writelsg, &ls);                     07896
             da.daccol _ rt.rtx1 _ tabcol - rt.rthinc;
             %convert fndtab return%                    07897
             END;                                       07898
          ENDCASE                                       07899
             BEGIN %make separate lsg with non-printing
             string %                                   07900
             %check if adding non-printing char string will
             exceed current line%                       07901
             bptr _ gapbp; %save old rtbpe value for lsg
             after non-printing char%                   07902
             IF 4 * da.dahinc + da.daccol > rt.rtx2 THEN
                                                        07903
                BEGIN                                   07904
                cdbckc(); %backup input text pointer%   07905
                rt.rtexis _ TRUE;                       07906
                IF trunc > 0 OR (NOT numflg OR (numflg AND
                da.daccol >= rt.rtx2 -
                (numberadr.L)*rt.rthinc)) THEN rt.rt1lsg _
                TRUE;                                   07907
                &ls _ nxtlsg(&da);                      07908
                rt.rtx1 _ da.daccol _ indent;           07909
                rt.rty _ da.dacrow := da.dacrow + da.davinc;
                                                        07910
                IF (da.dacrow > da.damrow) OR ( (trunc <= 0
                AND rt.rt1lsg) AND NOT (numflg OR
                viewspec1.vsidtf OR viewspec1.vsblkf) ) THEN
                ls.rtpartst _ FALSE; %statement now complete%
                                                        07911
                pushdc(writelsg, &ls);                  07912
                IF da.dacrow > da.damrow THEN EXIT LOOP;
                                                        07913
                END                                     07914
             ELSE                                       07915
                BEGIN                                   07916
```

```
                    &ls _ nxtlsg(&da); %get next line segment%
                                                              07917
                    ls.rtexis _ TRUE;                         07918
                    pushdc (writelsg, &ls); %write previous lsg%
                                                              07919
                    END;                                      07920
                  %set pointers to appropriate non-printing char
                  string%                                     07921
                  rt.rtbps _ chbmty + (&stringaddr _
                  npstrad(char));                             07922
                  rt.rtexis _ TRUE; %mark the entry as existing%
                                                              07923
                  gapbp _ chbptr (stringaddr.L) + &stringaddr;
                                                              07924
                  rt.rtcbug _ stringaddr.L; %make whole fat
                  character buggable%                         07925
                  da.daccol _  stringaddr.L * rt.rthinc + rt.rtx1;
                                                              07926
                  gapcol _ da.daccol - rt.rthinc;             07927
                  gapcc _ dgstl + 1; %advance past non-printing
                  char%                                       07928
                  &ls _ nxtlsg(&da);                          07929
                  rt.rtbps _ rt.rtbpe _ bptr; %restore old rtbpe
                  value pointing to user file%                07930
                  IF trunc <= 0 AND NOT(viewspec1.vsblkf OR
                  viewspec1.vsidtf OR numflg) THEN ls.rtpartst _
                  FALSE;                                      07931
                  pushdc (writelsg, &ls); %write previous lsg%
                                                              07932
                  END;                                        07933
            END;                                              07934
      END;                                                    07935
   %clear excess line segment%                                07936
      (stfrend):                                              07937
      rt.rtbpe _ rt.rtbps;                                    07938
   %***IF NOT cdvect() THEN RETURN(FALSE);***%                 07939
   IF numflg AND trunc <= 0 AND NOT (da.daccol >= da.damcol -
   rt.rthinc*(numberadr.L+1) AND da.dacrow > da.damrow) THEN   07940
      BEGIN %statement # on right at end of stmt unless both line
      and screen full%                                        07941
      numflg _ FALSE;                                         07942
      bptr _ rt.rtbps;                                        07943
      rt.rtbps _ chbmty + &numberadr;                         07944
      rt.rtbpe _ chbptr(numberadr.L) + &numberadr;            07945
      rt.rtsrce _ srcstno;                                    07946
      rt.rthinc _ da.danohi;                                  07947
      rt.rtcsize _ da.danocs;                                 07948
      char _ dgstl; %save old character count value used for
      sensing end of input%                                   07949
      rt.rtx2 _ da.damcol - da.dahinc;                        07950
      rt.rtx1 _ MAX(rt.rtx2 - (numberadr.L -1) * rt.rthinc, 0);
                                                              07951
      IF da.damcol < numberadr.L*rt.rthinc THEN               07952
         (setxc1):                                            07953
         *numberadr*[da.damcol /rt.rthinc] _ '!;              07954
      IF da.daccol >= rt.rtx1 THEN                            07955
```

```
      BEGIN  %have to put number on separate line%                    07956
      rt.rty _ da.dacrow := da.dacrow + da.davinc;                    07957
      blnklnf _ TRUE;                                                 07958
      END                                                             07959
   ELSE blnklnf _ FALSE;                                              07960
   IF NOT viewspec1.vsidtf THEN                                       07961
      BEGIN    %statement complete%                                   07962
      rt.rtpartst _ FALSE;                                            07963
      rt.rt1lsg _ TRUE;                                               07964
      END                                                             07965
   ELSE IF da.daccol < rt.rtx1 THEN rt.rt1lsg _ TRUE;                 07966
   gapcc _ 0;                                                         07967
   gapcol _ da.daccol _ da.damcol;                                    07968
   gapbp _ rt.rtbpe;                                                  07969
   &ls _ nxt1sg(&da);                                                 07970
   ls.rtexis _ TRUE;                                                  07971
   rt.rtx2 _ da.damcol - da.dahinc;                                   07972
   rt.rtx1 _ 0;                                                       07973
   rt.rt1lsg _ FALSE;                                                 07974
   rt.rtcsize _ da.dacsize;                                           07975
   rt.rthinc _ da.dahinc;                                             07976
   rt.rtbps _ rt.rtbpe _ bptr;                                        07977
   IF NOT (blnklnf AND viewspec1.vsidtf AND da.dacrow <=
   da.damrow) THEN pushdc(writelsg, &ls);      %May not really
   be able to put stmt # and signatures on same extra line;
   delay putting this line segment out until you find out
   unless the display is full%                                        07978
   rt.rtccnt _ (dgstl _ char) + 1; %restore old dgstl value -
   current char count%                                                07979
   END;                                                               07980
IF viewspec1.vsidtf AND NOT stid.stastr  THEN                         07981
   BEGIN                                                              07982
   IF da.dacrow <= da.damrow  THEN                                    07983
      BEGIN                                                           07984
      &stringaddr _ getstring(30, $dsphlk);                           07985
      rt.rtexis _ TRUE;                                               07986
      rt.rt1lsg _ TRUE;                                               07987
      IF NOT blnklnf THEN rt.rty _ da.dacrow := da.dacrow +
      da.davinc;                                                      07988
      *stringaddr* _ NULL;                                            07989
      fechsig(stid, &stringaddr);                                     07990
      rt.rtbps _ chbmty + &stringaddr;                                07991
      rt.rtbpe _ chbptr(stringaddr.L) + &stringaddr;                  07992
      rt.rtsrce _ srcsig;                                             07993
      rt.rthinc _ da.dasghi;                                          07994
      rt.rtcsize _ da.dasgcs;                                         07995
      IF blnklnf THEN                                                 07996
         BEGIN                                                        07997
         IF ls.rtx1 - (stringaddr.L+3)*ls.rthinc  < indent THEN
                                                                      07998
            BEGIN                                                     07999
            blnklnf _ FALSE;                                          08000
            ls.rt1lsg _ TRUE;                                         08001
            rt.rty _ da.dacrow := da.dacrow + da.davinc;              08002
            END;                                                      08003
         pushdc(writelsg, &ls);  %push out the stmt #%                08004
```

```
            END;                                                    08005
         IF blnklnf THEN rt.rtx2 _ da.daccol _ ls.rtx1 -
         (3*da.dahinc)                                              08006
         ELSE rt.rtx2 _ da.daccol _ da.damcol - da.dahinc;          08007
         rt.rtx1 _                                                  08008
            MAX(indent, rt.rtx2 - (stringaddr.L*rt.rthinc));        08009
         gapcc _ 0;                                                 08010
         gapcol _ rt.rtx2;                                          08011
         gapbp _ rt.rtbpe;                                          08012
         rt.rtpartst _ FALSE; %statement complete%                  08013
         &ls _ nxtlsg(&da);                                         08014
         pushdc(writelsg, &ls);  %push out the signature%           08015
         END;                                                       08016
      RETURN;                                                       08017
      END                                                           08018
   ELSE IF viewspec1.vsblkf AND NOT blnklnf THEN                    08019
      BEGIN                                                         08020
      IF da.dacrow <= da.damrow THEN                                08021
         BEGIN                                                      08022
         rt.rtexis _ TRUE;                                          08023
         rt.rtlsg _ TRUE;                                           08024
         rt.rty _ da.dacrow := da.dacrow + da.davinc;               08025
         rt.rtpartst _ FALSE; %statement complete%                  08026
         rt.rtsrce _ srcnul;                                        08027
         %clear line segment%                                       08028
            rt.rtbpe _ rt.rtbps _ chbmty;                           08029
            rt.rtx1 _ 0;                                            08030
         gapcol _ da.daright;                                       08031
         gapbp _chbmty;                                             08032
         &ls _ nxtlsg(&da);                                         08033
         pushdc(writelsg, &ls);                                     08034
         END;                                                       08035
      RETURN;                                                       08036
      END;                                                          08037
   %mkrsho();%                                                      08038
   IF trunc < 0 AND NOT viewspec1.vsidtf AND NOT viewspec1.vsblkf
   AND NOT numflg THEN ls.rtpartst _ FALSE;                         08039
   IF stid.stastr THEN                                              08040
      BEGIN   %must process commands accumulated so far to avoid
      losing the contents of the string%                            08041
      prcmds(&da);                                                  08042
      cmdinit($commands, &da); %zap the command list%               08043
      END;                                                          08044
   RETURN;                                                          08045
   END.                                                             08046
(lsgfrmt)  PROCEDURE (da);  %line segment formatter%                07450
   %Reasons for terminating a line segment:                         07451
      end of input: RETURN(ENDCHR),                                 07452
      end of allowed columns: RETURN(SP),                           07453
      CR was read: RETURN(CR),                                      07454
      TAB was read: RETURN(TAB),                                    07455
      control character was read: RETURN(the character),            07456
      font change: RETURN(swch1)%                                   07457
   %assumes globals dgstl, dgstml%                                  07458
   %-----------------%                                             07459
   LOCAL wrapgap, wrapcc, wrapbp;                                   07460
```

```
LOCAL bptr, char, col, hinc, x2lim, wwidth;                    07461
REF da;                                                        07462
rt.rtexis _ TRUE;                                              07463
%extract important variables%                                  07464
hinc _ rt.rthinc;                                              07465
x2lim _ (rt.rtx2/hinc*hinc) - (udpwrapcol*udpnwrap);           07466
wwidth _ (da.daright - da.daleft)/da.dahinc;                   07467
bptr _ rt.rtbpe;                                               07468
col _ da.daccol - hinc;                                        07469
wrapgap _ 0;                                                   07470
UNTIL (col _ col + hinc) > x2lim DO                            07471
   BEGIN                                                       07472
   IF (dgstl _ dgstl + 1) > dgstml THEN                        07473
      BEGIN                                                    07474
      (inpexh):  %input exhausted%                             07475
      gapbp _ bptr;                                            07476
      gapcol _ col - hinc; %last char of stmt not always
      erased%                                                  07477
      gapcc _ dgstl-1;                                         07478
      IF col = rt.rtx1 THEN                                    07479
         BEGIN %line segment is empty%                         07480
         %global lastch is used to determine if null line
         segment implies a null statement--a null statement's
         first char should be bugable%                         07481
         IF lastch NOT= ENDCHR THEN                            07482
            rt.rtsrce _ srcnul; %not a null statement--make it
            unbugable%                                         07483
         rt.rtx2 _ rt.rtx1; %only first char is bugable%       07484
         END;                                                  07485
      lastch _ char _ ENDCHR;                                  07486
      udpnwrap _ 0;                                            07487
      IF (wrapgap AND wrapgap # col - 1) THEN GOTO wrapout
                                                               07488
         ELSE GOTO lsfout;                                     07489
      END;                                                     07490
   CASE char _ ^bptr OF %get next character from statement%
                                                               07491
      IN ['a,'z], IN ['!,'_]:  NULL;                           07492
         %punctuation, lower case alpha, upper                 07493
         case alpha, numbers%                                  07494
      =SP:                                                     07495
         BEGIN %a gap character--note its position for line
         break%                                                07496
         IF (x2lim = wwidth AND col >= x2lim) THEN EXIT;       07497
         gapcol _ col;                                         07498
         gapcc _ dgstl;                                        07499
         gapbp _ bptr;                                         07500
         END;                                                  07501
      =TAB:                                                    07502
         BEGIN %ditto%                                         07503
         gapcol _ col;                                         07504
         gapcc _ dgstl;                                        07505
         gapbp _ bptr;                                         07506
         IF wrapgap THEN GOTO wrapout                          07507
            ELSE GOTO lsfout;                                  07508
         END;                                                  07509
```

```
      =CR, =EOL:                                                    07510
         BEGIN %ditto%                                              07511
         gapcol _ col;                                              07512
         gapcc _ dgstl;                                             07513
         gapbp _ bptr;                                              07514
         char _ CR;                                                 07515
         IF (wrapgap AND wrapgap # col - 1) THEN GOTO wrapout
                                                                    07517
            ELSE GOTO lsfout;                                       07518
         END;                                                       07519
      =CA, =C.,=0, =LF, =CD, =BC, =BW, =$ascalt, IN [1B,32B],
      IN [34B,36B]: %an acceptable non-printing character%
                                                                    07520
         BEGIN                                                      07521
         gapcol _ col - hinc; %back up 1 char - we have to
         terminate current lsg before handling non-printing
         char%                                                      07522
         gapcc _ dgstl - 1;                                         07523
         gapbp _ bptr;                                              07524
         IF wrapgap THEN GOTO wrapout                               07525
            ELSE GOTO lsfout;  %return the non-printing char%
                                                                    07526
         END;                                                       07527
      ENDCASE %some other character - will end up displaying as
      <NP>%                                                         07528
         REPEAT CASE(CA);                                           07529
   IF (wrapgap AND wrapgap < gapcol) THEN EXIT LOOP;                07530
   IF col + 1 = udpwrapcol THEN                                     07531
      BEGIN  %note where to wrap%                                   07532
      wrapgap _ col;                                                07533
      wrapcc _ dgstl;                                               07534
      wrapbp _ bptr;                                                07535
      END;                                                          07536
   %IF mkrflg THEN mkrset(char);% %should reinstitute marker
   display someday%                                                 07537
   END;  % of loop:  UNTIL (col _ col + hinc) ... %                 07538
%exceeded allowed length of line segment%                           07539
IF (gapcol = 0 AND wrapgap = 0) THEN                                07540
   BEGIN %no gap char found%                                        07541
   gapcol _ col - hinc; %make current char be a gap char%           07542
   gapcc _ dgstl;                                                   07543
   gapbp _ bptr;                                                    07544
   udpnwrap _ 0;                                                    07545
   IF dgstl = 0 THEN                                                07546
      char _ ENDCHR %no characters read yet%                        07547
   ELSE                                                             07548
      BEGIN                                                         07549
      rt.rtbpe _ bptr;                                              07550
      da.daccol _ col-hinc;                                         07551
      cdbckc(); %back up input one char%                            07552
      gapbp _ rt.rtbpe;                                             07553
      gapcol _ gapcol - hinc;                                       07554
      gapcc _ gapcc - 1;                                            07555
      RETURN(SP); %end of line%                                     07556
      END;                                                          07557
   END                                                              07558
```

```
      ELSE                                                              07559
         BEGIN                                                          07560
         IF ((gapcol = x2lim AND wrapgap = 0) AND (char = SP AND
         dgstl < dgstml)) THEN                                          07561
            BEGIN                                                       07562
            col _ col - hinc;                                           07563
            WHILE (x2lim _ x2lim + hinc) < MIN(wwidth, IF udpwrapcol
            THEN udpwrapcol ELSE 1000) DO                               07564
               BEGIN                                                    07565
               IF (dgstl _ dgstl + 1) > dgstml THEN                     07566
                  BEGIN                                                 07567
                  col _ col + hinc;                                     07568
                  GOTO inpexh;                                          07569
                  END;                                                  07570
               CASE char _ ^bptr OF                                     07571
                  =SP, =TAB:                                            07572
                     BEGIN                                              07573
                     gapcol _ col _ col + hinc;                         07574
                     gapcc _ dgstl;                                     07575
                     gapbp _ bptr;                                      07576
                     IF char = TAB THEN GOTO lsfout;                    07577
                     END;                                               07578
                  =EOL, =CR:                                            07579
                     BEGIN                                              07580
                     gapcol _ col _ col + hinc;                         07581
                     gapcc _ dgstl;                                     07582
                     gapbp _ bptr;                                      07583
                     char _ CR;                                         07584
                     IF dgstl = dgstml THEN REPEAT LOOP;                07585
                     GOTO lsfout;                                       07586
                     END;                                               07587
                  ENDCASE                                               07588
                     BEGIN                                              07589
                     cdbckc(); %back up input one char%                 07590
                     dgstl _ dgstl - 1;                                 07591
                     EXIT LOOP;                                         07592
                     END;                                               07593
               END;                                                     07594
            udpnwrap _ 0;                                               07595
            END;                                                        07596
         IF (wrapgap AND (wrapgap < gapcol OR gapcol = 0)) THEN         07597
            BEGIN                                                       07598
            (wrapout):                                                  07599
            udpnwrap _ udpnwrap + 1;                                    07600
            da.daccol _ gapcol _ wrapgap;                               07601
            rt.rtbpe _ gapbp _ wrapbp;                                  07602
            dgstl _ gapcc _ wrapcc;                                     07603
            RETURN(SP);                                                 07604
            END;                                                        07605
         da.daccol _ gapcol;                                            07606
         rt.rtbpe _ gapbp;                                              07607
         dgstl _ gapcc;                                                 07608
         udpnwrap _ 0;                                                  07609
         RETURN(SP);                                                    07610
         END;                                                           07611
      (lsfout): %restore extracted variables%                           07612
```

```
      rt.rtbpe _ bptr;                                                  07613
      da.daccol _ col;                                                  07614
      RETURN(char);                                                     07615
      END.                                                              07616
                                                                        07617
   (nxtlsg)   PROCEDURE (da);  %line segment termination%               05436
      %terminate current line segment and allocate a new one,           05437
      updating RT, and moving the excess characters to the              05438
      new line segment.  Allocate new LSRT if space permits.%           05439
      %-----------------%                                               05440
      LOCAL                                                             05441
         char, ls, save, oldtable;                                      05442
      REF ls, da;                                                       05443
      IF &rt >= rte THEN                                                05444
         BEGIN                                                          05445
         dismes(1, $"DISPLAY ERROR - attempting recovery...
         screen may be incomplete");                                    05446
         &rt _ &rt - lsrtl; %one segment may be thrown away%            05447
         END;                                                           05448
      oldtable _ FALSE;                                                 05449
      rt.rtbpe _ gapbp;                                                 05450
      save _ rt.rtx2;                                                   05451
      rt.rtx2 _ gapcol;                                                 05452
      rt.rtnew _ TRUE;                                                  05453
      &ls _ &rt := &rt + lsrtl;                                         05454
      IF &rt >= rte THEN                                                05455
         BEGIN                                                          05456
         oldtable _ dspace(10);%increase allocation 10 entries%         05457
         IF NOT oldtable THEN %failed:  either request exceeded max
         block size, or we are really out of space in the display
         block%                                                         05458
            BEGIN                                                       05459
            &rt _ &rt - lsrtl; %don't allow increment - dangerous%
                                                                        05460
            IF (lsrtl * (rtsize + 10)) < 512 %requested less than
            max% AND da.dacrow < (da.damrow - da.davinc) THEN           05461
               dismes(1, $"System had to truncate display.");           05462
            SIGNAL (spacerr, $" ");                                     05463
            END;                                                        05464
         END;                                                           05465
      dgstl _ gapcc;                                                    05466
      %initialize next LSRT entry%                                      05467
         rt.rtx1 _ ls.rtx2 + ls.rthinc;                                 05468
         rt.rty _ ls.rty;                                               05469
         rt.rtpartst _ TRUE;                                            05470
         rt.rtsrce _ ls.rtsrce;                                         05471
         rt.rtfont _ ls.rtfont;                                         05472
         rt.rtstid _ ls.rtstid;                                         05473
         rt.rtlev _ ls.rtlev;                                           05474
         rt.rthinc _ ls.rthinc;                                         05475
         rt.rtcsize _ ls.rtcsize;                                       05476
         rt.rtcbug _ 1;                                                 05477
         rt.rtexis _ rt.rtllsg _ FALSE;                                 05478
         rt.rtnew _ TRUE;                                               05479
         rt.rtbps _ rt.rtbpe _ gapbp;                                   05480
         rt.rtccnt _ gapcc + 1;                                         05481
```

```
      rt.rtx2 _ save;                                                  05482
      rt.rtlsid _ 0;                                                   05483
   gapcol _ gapcc _ 0;                                                 05484
   %now its safe to free old table%                                    05485
      IF oldtable %had to reallocate% THEN                             05486
         freeblk(oldtable -  bhl, $dspblk);                            05487
   RETURN(&rt - lsrtl);                                                05488
   END.                                                                05489
                                                                       05490
(dspace) PROCEDURE(addition);                                           0942
   %allocate addition more lsrtl entries for table at rttop,
   updating globals rttop, rtsize, rt, rte%                             0943
   %returns FALSE if failed, address of old table if success,
   because some callers like to free their own table later on%
                                                                        0944
   LOCAL newtable, oldtable;                                            0945
   IF NOT (newtable _ getblk ((rtsize + addition) * lsrtl,
   $dspblk)) THEN                                                       0946
      RETURN(FALSE);                                                    0947
   %copy old to new%                                                    0948
      newtable _ newtable + bhl; %start of useable storage%             0949
      mvbfbf(rttop, newtable, rtsize * lsrtl);                          0950
      oldtable _ rttop; %need for freeing storage%                      0951
   %update global addresses%                                            0952
      rttop _ newtable;                                                 0953
      &rt _ (newtable + rtsize * lsrtl);                                0954
      rte _ &rt + (addition * lsrtl);% new end + 1%                     0955
      rtsize _ rtsize + addition;                                       0956
      RETURN(oldtable);                                                 0957
      END.                                                              0958
                                                                        0959
(cdbckc)   PROCEDURE;  %character output routine%                       0960
   IF rt.rtbpe.bpadr = rt.rtbps.bpadr - 1 THEN                          0961
   RETURN(ENDCHR);                                                      0962
   IF rt.rtbpe.bpbitpos <= 29 THEN                                      0963
      rt.rtbpe.bpbitpos _ rt.rtbpe.bpbitpos + 7                         0964
   ELSE                                                                 0965
      BEGIN                                                             0966
      rt.rtbpe.bpbitpos _ 1;                                            0967
      BUMP DOWN rt.rtbpe.bpadr;                                         0968
      END;                                                              0969
   RETURN(.rt.rtbpe);                                                   0970
   END.                                                                 0971
                                                                        0972
(npstrad) PROCEDURE (npchar);%get symbol for non-printing%              0973
   %npstrad accepts a non-printing character code, npchar, as
   input, and returns the address of the appropriate global string
   used to represent that character in the display%                     0974
   LOCAL index;  %for matching npcodes and npstrings%                   0975
      FOR index _ 0 UP 1 UNTIL >= npsize DO                             0976
         IF npcodes[index] = npchar THEN RETURN
         (npstrings[index]);                                            0977
      RETURN ($unknown);                                                0978
   END.                                                                 0979
                                                                        0980
(srch) PROCEDURE (stid, level, match, top);%find already formatted
```

```
   stid%                                                              0981
      %given an STID and level, srch finds a match, if possible, in
      the LSRT, using top as the starting address of the search.
      Returns match = TRUE if found, and address of new top, if top
      may be incremented.%                                            0982
      LOCAL save;                                                     0983
      % check if top out of allowable range %                         0984
         IF (&art _  top) >= arte OR NOT art.rtexis THEN              0985
            RETURN (FALSE,FALSE);                                     0986
      % increment table top if match on first entry%                  0987
         IF art.rtstid = stid AND art.rtlev = level THEN              0988
            BEGIN                                                     0989
            IF ([(save _ lstidnxt(&art)) - lsrtl].rtpartst)  AND
            (art.rty NOT= rt.rty) THEN                                0990
               RETURN (FALSE, save) %partial stmt%                    0991
            ELSE                                                      0992
               RETURN (TRUE, save);                                   0993
            END;                                                      0994
      % main search loop%                                            0995
      &art _ lstidnxt(&art); %advance ptr to next stid%               0996
      UNTIL NOT art.rtexis OR &art >= arte CR ([(save _
      lstidnxt(&art)) - lsrtl].rtpartst AND (art.rty NOT= rt.rty))
      %partial stmt% DO                                               0997
         BEGIN                                                        0998
         IF art.rtstid = stid AND art.rtlev = level THEN %match%      0999
            BEGIN %check for special case, jumps%                    01000
            IF cdtype = dspjpf THEN RETURN (TRUE, save)              01001
            ELSE RETURN (TRUE, FALSE);                               01002
            END;                                                     01003
         &art _ lstidnxt (&art); %advance to next stid%              01004
         END;                                                        01005
      RETURN (FALSE,FALSE); %no match found%                         01006
      END.                                                           01007
                                                                     01008
   (lstidnxt) PROCEDURE (start) ; %next stid's LSRT address%         01009
      %given an lsrt entry address, return the address of the first
      entry for the statement after this one or end of table + 1%
                                                                     01010
      UNTIL start >= (arte - lsrtl) OR NOT [start].rtexis DO         01011
         IF [start].rtstid # [(start _ start + lsrtl)].rtstid THEN
                                                                     01012
            RETURN (start);                                          01013
      RETURN (start); %hit end of table - return illegal address%
                                                                     01014
      END.                                                           01015
                                                                     01016
   (lscopy) PROCEDURE (option,da);%copy LSRT entries%                01017
      %option = TRUE if only one statement to copy - else whole rest
      of &art area is copied to &rt area%                            01018
         %the big hangup with this routine is the possibility of
         running into a partial statement and having to back up%
                                                                     01019
                                                                     01020
      LOCAL oldrt, rtoffset, oldrow, lastid, oldtable;               01021
      REF da;                                                        01022
                                                                     01023
```

```
%check for no source data%                                          01024
   IF da.dacrow > da.damrow OR NOT art.rtexis OR &art >= arte
   THEN RETURN(FALSE);                                              01025
%initialize locals%                                                 01026
   oldrt _ &rt;                                                     01027
   oldrow _ da.dacrow;                                              01028
   lastid _ art.rtstid; %stid of first source%                      01029
   oldtable _ TRUE;                                                 01030
%segment by segment copy loop%                                      01031
   DO                                                               01032
      BEGIN                                                         01033
      lsmove(); %move one segment across to &rt%                    01034
      rt.rty _ da.dacrow;                                           01035
      IF rt.rtllsg THEN %last segment in the line%                  01036
         da.dacrow := da.dacrow + da.davinc;                        01037
      &rt _ &rt + lsrtl; %increment destination%                    01038
      %check for destination overflow%                              01039
         IF &rt >= rte THEN %get more space%                        01040
            BEGIN                                                   01041
            rtoffset _ &rt - oldrt; %offset between backup
            factor and current destination addr%                    01042
            oldtable _ dspace(10);                                  01043
            IF NOT oldtable THEN                                    01044
               BEGIN                                                01045
               dismes(2,$"Non-fatal System error - short on
               space");                                             01046
               EXIT LOOP;%allocate failed%                          01047
               END;                                                 01048
            freeblk(oldtable - bhl, $dspblk);                       01049
            oldrt _ &rt - rtoffset; %whole table relocated%
                                                                    01050
            END;                                                    01051
      %update backup variables in case next one incomplete%
                                                                    01052
         IF NOT [&rt-lsrtl].rtpartst %previous seg complete%
         THEN                                                       01053
            BEGIN                                                   01054
            oldrt _ &rt;                                            01055
            oldrow _ da.dacrow;                                     01056
            END;                                                    01057
      END %of copy loop%                                            01058
   UNTIL (option AND art.rtstid # lastid) %done with requested
   1 stmt copy%
   OR da.dacrow > da.damrow %filled screen%
   OR NOT art.rtexis OR &art >= arte %no more source data%
   OR  &rt >= rte % destination overflow%;                          01059
IF oldtable AND
NOT ([&rt-lsrtl].rtpartst AND ([&rt-lsrtl].rty NOT=
[&art-lsrtl].rty)) THEN RETURN(TRUE);                               01060
                                                                    01061
%bottom of screen - back up y-cord and address to end of last
good copy%                                                          01062
   &rt _ oldrt;                                                     01063
   da.dacrow _ oldrow;                                              01064
   RETURN(FALSE);                                                   01065
END.                                                                01066
```

```
                                                                    01067
   (complsg) PROCEDURE (bps1, bps2, length);  %compare two line
   segments -- Return TRUE is they contain the same string of
   characters%                                                      01068
      FOR length DOWN UNTIL <= 0 DO                                 01069
         IF ^bps1 NOT= ^bps2 THEN RETURN (FALSE);                   01070
      RETURN( TRUE );                                               01071
      END.                                                          01072
                                                                    01073
   (lsmove) PROCEDURE;                                              01074
      %move line segment entry from &art to &rt, leaving only &art
      updated%                                                      01075
      LOCAL end, ls;                                                01076
      REF ls;                                                       01077
      end _ (&ls _ &rt) + lsrtl;                                    01078
      DO                                                            01079
         BEGIN                                                      01080
         ls _ art;                                                  01081
         BUMP &art;                                                 01082
         END                                                        01083
      UNTIL (&ls _ &ls + 1) = end;                                  01084
      rt.rtnew _ FALSE;                                             01085
      RETURN END.                                                   01086
                                                                    01087
                                                                    01088
%** leave pointers out for initial system **%                       01089
   (mkrsho) PROCEDURE;                                              01090
      RETURN;                                                       01091
      %****%                                                        01092
      END.                                                          01093
                                                                    01094
   (mkrset) PROCEDURE;                                              01095
       %col # is da.daccol %                                        01096
       %swork is current psid %                                     01097
       %ptrtb is pointer table %                                    01098
       %ptrtbl is max length of the table %                         01099
       %mkrptr is addr in ptrtb of the next possible                01100
      pointer%                                                      01101
       %ptrse2 points to loc in pointer buffer%                     01102
       % mkrend is address of end of legal pointers in table       01103
      %                                                             01104
       % mkrcnt is character number in statement of next           01105
      pointer %                                                     01106
      %-----------------%                                          01107
      %******%                                                     01108
       RETURN;                                                      01109
      %******%                                                     01110
      END.                                                          01111
                                                                    01112
%...display command list handling....%                             01113
   (pushdc) PROCEDURE (op, parm1);                                  01114
      %enter commands into the display commands list%               01115
      LOCAL entry,%address temp%                                    01116
         da,      %display area address%                          01117
         curblk;  %current block being filled%                     01118
      REF curblk, entry, da;                                        01119
```

```
    &curblk _ dpcmblk; %current block in use%                          01120
    &da _ curblk.cmda;                                                 01121
    CASE &entry _ (curblk.cmnxt := curblk.cmnxt + 1) OF                01122
       <= curblk.cmlst:   %room in this block%                         01123
          BEGIN                                                        01124
          entry.copcode _ op;                                          01125
          entry.cparm1 _ parm1 - rttop; %make LSRT entry address
          relative in case allocations change%                         01126
          entry.ctype _ FALSE;                                         01127
          END;                                                         01128
       ENDCASE   %no room in this block%                               01129
          BEGIN                                                        01130
          IF dpcmblk _ &entry _ curblk.cbnxt %next block exists
          physically, not logically%                                   01131
          THEN GO TO newblk;                                           01132
          dpcmblk _ getblk (65, $dspblk); %allocate new block%         01133
          IF NOT dpcmblk THEN                                          01134
             BEGIN                                                     01135
             dismes(1,$"FATAL DISPLAY ERROR ^C and RESET
             Reenter NLS by retyping 'NLS'.");                         01136
             SIGNAL(spacerr, $"FATAL DISPLAY ERROR - type ^C then
             RESET
             Reenter NLS by retyping 'NLS'"); %he's in trouble%        01137
             END;                                                      01138
          curblk.cbnxt _ &entry _ dpcmblk _ (dpcmblk + bhl); %block
          header%                                                      01139
          entry.cbprev _ &curblk;                                      01140
          entry.cmlst _ &entry + 64 - cbhl;                            01141
          entry.cbnxt _ 0;                                             01142
          entry.cmda _ &da; %same as last block%                       01143
       (newblk):                                                       01144
          &curblk _ &entry;                                            01145
          entry.cmnxt _ &entry + cbhl;                                 01146
          &entry _ (entry.cmnxt := entry.cmnxt + 1);                   01147
          REPEAT CASE (&entry);                                        01148
          END;                                                         01149
       RETURN END.                                                     01150
                                                                       01151
(prcmds) PROCEDURE (da); %post processor%                              09018
    LOCAL mtop, wtop, dtop, mi, wi, di, ls, dls, mls, linkls, dlinkls,
    base, end, line, nline, tdline, pdabottom, pdaright, entry,
    savblk, savnxt, writend, curblk, clear, replace, noop, ms[150],
    ws[150], ds[150];                                                  09019
    REF art, da, ls, dls, mls, entry, savblk, linkls, dlinkls;         09020
    %initialization%                                                   09021
    curblk _ dpcmblk; %last block written%                             09022
    &entry _ [curblk].cmnxt; %addr + 1 of last used entry%             09023
    %main processing loop%                                             09024
    CASE nldevice OF                                                   09025
       =devlproc: %alpha-numeric display%                              09026
          BEGIN                                                        09027
          %initialize%                                                 09028
             IF &da THEN                                               09029
                BEGIN                                                  09030
                pdabottom _ da.dabottom;                               09031
                IF pdabottom # lpymax THEN BUMP DOWN pdabottom;        09032
```

```
      pdaright _ da.daright;                                    09536
      IF pdaright # lpxmax THEN BUMP DOWN pdaright;             09537
      END;                                                      09033
   clear _ 100; replace _ 101; noop _ 102;                      09034
      %internal values for copcode%                             09035
   mtop _ $ms; wtop _ $ws; dtop _ $ds;                          09036
UNTIL NOT cnxtaddress(&entry, curblk, backward  :&entry,
curblk) DO                                                      09037
   CASE entry.copcode OF                                        09038
      =writelsg:                                                09039
         BEGIN                                                  09040
         &ls _ entry.cparm1 + rttop;                            09041
         IF ls.rtnew THEN                                       09042
            [wtop := wtop + 1] _ &entry;                        09043
         END;                                                   09044
      =movelsg: [mtop := mtop + 1] _ &entry;                    09045
      =deletlsg: [dtop := dtop + 1] _ &entry;                   09046
      =blankda:  %process now%                                  09047
         clearda(&da);                                          09048
      =suppressda: %process now%                                09049
         supda(&da);                                            09050
      =restoreda: %process now%                                 09051
         resda(&da);                                            09052
      ENDCASE err($"undefined display command");                09053
(prcmd1):                                                       09054
%check for several writes/deletes/moves to same line -- link
them together%                                                  09055
   FOR wi _ wtop - 1 DOWN UNTIL < $ws DO                        09056
      BEGIN                                                     09057
      &ls _ [[wi]].cparm1 + rttop;                              09058
      IF NOT ls.rtlplink AND NOT ls.rtllsg THEN %look for
      others on same line%                                      09059
         BEGIN                                                  09060
         line _ ls.rty;                                         09061
         &dls _ &ls;                                            09062
         FOR mi _ wi - 1 DOWN UNTIL < $ws DO                    09063
            BEGIN                                               09064
            &mls _ [[mi]].cparm1 + rttop;                       09065
            IF mls.rty = line AND dls.rtx2 < mls.rtx1 THEN
                                                                09066
               BEGIN                                            09067
               dls.rtlplink _ &mls - &dls;                      09068
               &dls _ &mls;                                     09069
               [[mi]].copcode _ noop;                           09070
               END;                                             09071
            END;                                                09072
         END;                                                   09073
      END;                                                      09074
   FOR di _ dtop - 1 DOWN UNTIL < $ds DO                        09075
      BEGIN                                                     09076
      &ls _ [[di]].cparm1 + rttop;                              09077
      IF NOT ls.rtlplink AND NOT ls.rtllsg THEN %look for
      others on same line%                                      09078
         BEGIN                                                  09079
         line _ ls.rty;                                         09080
         &dls _ &ls;                                            09081
```

```
            FOR mi _ di - 1 DOWN UNTIL < $ds DO                    09082
               BEGIN                                               09083
               &mls _ [[mi]].cparm1 + rttop;                       09084
               IF mls.rty = line AND dls.rtx2 < mls.rtx1 THEN
                                                                   09085
                  BEGIN                                            09086
                  dls.rtlplink _ &mls - &dls;                      09087
                  &dls _ &mls;                                     09088
                  [[mi]].copcode _ noop;                           09089
                  END;                                             09090
               END;                                                09091
            END;                                                   09092
         END;                                                      09093
      FOR mi _ mtop - 1 DOWN UNTIL < $ms DO                        09094
         BEGIN                                                     09095
         &ls _ [[mi]].cparm1 + rttop;                              09096
         IF NOT ls.rtlplink AND NOT ls.rtllsg THEN %look for
         others on same line%                                      09097
            BEGIN                                                  09098
            line _ ls.rty;                                         09099
            &dls _ &ls;                                            09100
            FOR wi _ mi - 1 DOWN UNTIL < $ms DO                    09101
               BEGIN                                               09102
               &mls _ [[wi]].cparm1 + rttop;                       09103
               IF mls.rty = line AND dls.rtx2 < mls.rtx1 THEN
                                                                   09104
                  BEGIN                                            09105
                  dls.rtlplink _ &mls - &dls;                      09106
                  &dls _ &mls;                                     09107
                  [[wi]].copcode _ noop;                           09108
                  END;                                             09109
               END;                                                09110
            END;                                                   09111
         END;                                                      09112
      (prcmd2):                                                    09113
      CASE (IF mtop = $ms THEN 0 ELSE 4) .V (IF wtop = $ws THEN 0
      ELSE 2) .V (IF dtop = $ds THEN 0 ELSE 1) OF                  09114
         = 1: %deletes only - no moves or writes%                  09115
            BEGIN %process all deletes as clears%                  09116
            (prcmd3):                                              09117
            FOR di _ $ds UP UNTIL >= dtop DO                       09118
               IF [[di]].copcode = deletlsg THEN                   09119
                  BEGIN                                            09120
                  &dls _ &ls _ [[di]].cparm1 + rttop;              09121
                  LOOP                                             09122
                     BEGIN                                         09123
                     dls.rtlsid _ 0;                               09124
                     IF dls.rtlplink THEN &dls _ &dls +
                     dls.rtlplink                                  09125
                     ELSE EXIT LOOP;                               09126
                     END;                                          09127
                  cline(ls.rtx1+da.daleft, ls.rty+da.datop,
                  (dls.rtx2-ls.rtx1)/ls.rthinc + 1);               09128
                  END;                                             09129
            END;                                                   09130
         = 2: %writes only - no moves or deletes%                  09131
```

```
        BEGIN                                                   09132
        (prcmd4):                                               09133
        % process writes as replaces%                           09134
        FOR wi _ wtop-1 DOWN UNTIL < $ws DO                     09135
           IF [[wi]].copcode = writelsg THEN                    09136
              BEGIN                                             09137
              &ls _ [[wi]].cparm1 + rttop;                      09138
              IF ls.rtx1 > 0  AND (lptype .A 17B) = deltadata
              THEN %must pad for display delay%                 09139
                 cline(da.daleft, ls.rty+da.datop,
                 ls.rtx1/ls.rthinc);                            09140
              wrtstr(&da, &ls);                                 09141
              ls.rtlsid _ da.dahandle := da.dahandle+1;         09142
              WHILE ls.rtlplink DO                              09143
                 BEGIN                                          09144
                 &ls _ &ls + ls.rtlplink;                       09145
                 ls.rtlsid _ da.dahandle := da.dahandle+1;
                                                                09146
                 END;                                           09147
              END;                                              09148
        END;                                                    09149
     IN [3,7]: % writes, deletes, and moves%                    09150
        BEGIN                                                   09151
        (prcmd5):                                               09152
        % For each write, if there is a delete for the same
        line  and no moves FROM/TO above it then make the
        write a replace and the delete a clear (if old line is
        farther to left than new line) otherwise a noop.
        Otherwise, process writes as inserts and deletes as
        deletes%                                                09153
        IF dtop > $ds THEN                                      09154
           BEGIN                                                09155
           FOR wi _ wtop-1 DOWN UNTIL < $ws DO                  09156
              IF [[wi]].copcode = writelsg THEN                 09157
                 BEGIN                                          09158
                 &ls _ [[wi]].cparm1 + rttop;                   09159
                 (prcmd6):                                      09160
                 ls.rtlsid _ ls.rtx2; %save rtx2 -- it may get
                 smashed%                                       09161
                 FOR di _ $ds UP UNTIL >= dtop DO               09162
                    IF [[di]].copcode = deletlsg THEN           09163
                       BEGIN                                    09164
                       &dls _ [[di]].cparm1 + rttop;            09165
                       (prcmd7):                                09166
                       IF ls.rty = dls.rty THEN %there is a
                       write and delete to same line%           09167
                          BEGIN                                 09168
                          end _ TRUE;                           09169
                          %determine if any moves to/from
                          above this line%                      09170
                             arte _ oldlsrt + (lsrtl *
                             da.dalsz);                         09171
                             FOR mi _ $ms UP UNTIL >= mtop DO
                                                                09172
                                IF [[mi]].copcode = movelsg
                                THEN                            09173
```

```
            BEGIN                          09174
            &mls _ [[mi]].cparm1 +
            rttop;                         09175
            IF mls.rty <= ls.rty THEN
                                           09176
            BEGIN %a line is being
            moved to above this one,
            thus delete must delete the
            line and write must insert
            new line%                      09177
            end _ FALSE;                   09178
            EXIT;                          09179
            END;                           09180
            %otherwise must see if
            moving from above this
            line%                          09181
            FOR &art _ oldlsrt UP lsrtl
            UNTIL >= arte DO               09182
            IF art.rtexis AND
            art.rtlsid = mls.rtlsid
            THEN                           09183
            EXIT;                          09184
            IF &art < arte AND art.rty
            <= ls.rty THEN                 09185
            BEGIN %a line is being
            moved from above this one,
            thus delete must delete the
            line and write must insert
            new line%                      09186
            end _ FALSE;                   09187
            EXIT;                          09188
            END;                           09189
            END;                           09190
IF end THEN %no moves to/from above
this line%                                 09191
   BEGIN                                   09192
   (prcmd8):                               09193
   [[wi]].copcode _ replace;               09194
   [[di]].copcode _ noop;                  09195
   %check for need to write trailing
   blanks%                                 09196
      &dlinkls _ &dls;                     09197
      IF dls.rtlplink THEN                 09198
         WHILE dlinkls.rtlplink DO
         &dlinkls _ &dlinkls +
         dlinkls.rtlplink;                 09199
      &linkls _ &ls;                       09200
      IF ls.rtlplink THEN                  09201
         WHILE linkls.rtlplink DO
         &linkls _ &linkls +
         linkls.rtlplink;                  09202
      IF dlinkls.rtx2 > linkls.rtx2
      THEN % must write trailing
      blanks%                              09203
         linkls.rtx2 _ dlinkls.rtx2;
                                           09204
```

```
                        IF dls.rtx1 < ls.rtx1 THEN %must
                        write begining blanks%          09205
                           BEGIN                        09206
                           [[di]].copcode _ clear;      09207
                           dls.rtx2 _ ls.rtx1;          09208
                           END;                         09209
                        END;                            09210
                     END;                               09211
                  END;                                  09212
               END;                                     09213
            END;                                        09214
         %moves -- erase old instances of these line segments%
                                                        09215
            FOR mi _ mtop-1 DOWN UNTIL < $ms DO         09216
               IF [[mi]].copcode = movelsg THEN         09217
                  BEGIN                                 09218
                  (prcm30):                             09219
                  &mls _ [[mi]].cparm1 + rttop;         09220
                  %determine if moved up or down correct number
                  of lines lines%                       09221
                     arte _ oldlsrt + (lsrtl * da.dalsz); 09222
                     FOR &art _ oldlsrt UP lsrtl UNTIL >= arte
                     DO                                 09223
                        IF art.rtexis AND art.rtlsid =
                        mls.rtlsid THEN                 09224
                           EXIT;                        09225
                     (prcm31):                          09226
                     IF &art < arte THEN                09227
                        BEGIN                           09228
                        line _ [[mi]].cparm2 _ art.rty;
                        %y[old]%                        09229
                        IF scwindow OR (NOT da.dalneighbor AND
                        NOT da.darneighbor AND NOT
                        da.datneighbor AND NOT da.dabneighbor)
                        THEN                            09230
                           BEGIN                        09231
                           (prcm32):                    09232
                           FOR di _ $ds UP UNTIL >= dtop DO
                                                        09233
                              IF [[di]].copcode = deletlsg THEN
                                                        09234
                                 BEGIN                  09235
                                 &dls _ [[di]].cparm1 + rttop;
                                                        09236
                                 IF dls.rty < line THEN %line
                                 is moving up%          09237
                                    BEGIN               09238
                                    line _ line - da.davinc;
                                                        09239
                                    END;                09240
                                 END;                   09241
                           FOR wi _ wtop-1 DOWN UNTIL < $ws DO
                                                        09242
                              IF [[wi]].copcode = writelsg THEN
                                                        09243
                                 BEGIN                  09244
```

```
                                &ls _ [[wi]].cparm1 + rttop;
                                                                09245
                                IF ls.rty <= line THEN          09246
                                   BEGIN %line is moving down%
                                                                09247
                                   line _ line + da.davinc;
                                                                09248
                                   END;                         09249
                                END;                            09250
                             IF line = mls.rty THEN
                             [[mi]].copcode _ noop;             09251
                             END;                               09252
                          END                                   09253
                       ELSE %what to do???% [[mi]].copcode _
                       noop;                                    09254
                    END;                                        09255
                %clear lines where line segments were%          09256
                   FOR mi _ mtop-1 DOWN UNTIL < $ms DO          09257
                      IF [[mi]].copcode = movelsg THEN          09258
                         BEGIN                                  09259
                         &ls _ [[mi]].cparm1 + rttop;           09260
                         (prcm33):                              09261
                         % do a cline where this line segment was.%
                                                                09262
                            cline(da.daleft,
                            [[mi]].cparm2+da.datop,
                            (da.daright-da.daleft)/da.dahinc);
                                                                09263
                         END;                                   09264
             %clears%                                           09265
                FOR di _ $ds UP UNTIL >= dtop DO                09266
                   IF [[di]].copcode = clear THEN               09267
                      BEGIN                                     09268
                      &ls _ [[di]].cparm1 + rttop;              09269
                      LOOP                                      09270
                         BEGIN                                  09271
                         (prcmd9):                              09272
                         ls.rtlsid _ 0;                         09273
                         cline(ls.rtx1+da.daleft, ls.rty+da.datop,
                         (ls.rtx2-ls.rtx1)/ls.rthinc + 1);      09274
                         IF ls.rtlplink THEN                    09275
                            &ls _ &ls + ls.rtlplink             09276
                         ELSE EXIT LOOP;                        09277
                         END;                                   09278
                      END;                                      09279
             %replaces%                                         09280
                FOR wi _ wtop-1 DOWN UNTIL < $ws DO             09281
                   IF [[wi]].copcode = replace THEN             09282
                      BEGIN                                     09283
                      &ls _ [[wi]].cparm1 + rttop;              09284
                      (prcm11):                                 09285
                      IF ls.rtx1 > 0 AND (lptype .A 17B) =
                      deltadata THEN %must pad for display delay%
                                                                09286
                         cline(da.daleft, ls.rty+da.datop,
                         ls.rtx1);                              09287
```

```
                wrtstr(&da, &ls);                                   09288
                ls.rtx2 _ ls.rtlsid; %restore rtx2%                 09289
                ls.rtlsid _ da.dahandle := da.dahandle+1;
                                                                    09290
                WHILE ls.rtlplink DO                                09291
                   BEGIN                                            09292
                   &ls _ &ls + ls.rtlplink;                         09293
                   ls.rtlsid _ da.dahandle := da.dahandle+1;
                                                                    09294
                   END;                                             09295
                END;                                                09296
          %deletes%                                                 09297
             IF scwindow THEN                                       09298
                BEGIN                                               09299
                di _ $ds; %do scrolls instead of deletes%           09300
                WHILE di < dtop DO                                  09301
                   BEGIN %for each contiguous group of deletes%
                                                                    09302
                   nline _ 0;                                       09303
                   WHILE di < dtop DO                               09304
                      BEGIN                                         09305
                      IF [[di]].copcode = deletlsg THEN             09306
                         BEGIN                                      09307
                         &ls _ [[di]].cparm1 + rttop;               09308
                         line _ ls.rty;                             09309
                         IF nline=0 OR line=tdline-1 THEN           09310
                            BEGIN                                   09311
                            tdline _ line;                          09312
                            BUMP nline;                             09313
                            END                                     09314
                         ELSE EXIT;                                 09315
                         END;                                       09316
                      BUMP di;                                      09317
                      END;                                          09318
                   IF nline#0 THEN doscroll(da.daleft, pdaright,
                   da.datop+tdline, pdabottom, nline);              09319
                   END;                                             09320
                END                                                 09321
             ELSE FOR di _ $ds UP UNTIL >= dtop DO                  09322
                IF [[di]].copcode = deletlsg THEN                   09323
                   BEGIN                                            09324
                   &ls _ [[di]].cparm1 + rttop;                     09325
                   (prcm12):                                        09326
                   ls.rtlsid _ 0;                                   09327
                   IF da.dalneighbor OR da.darneighbor OR
                   da.datneighbor OR da.dabneighbor THEN %has a
                   left, right , top, or bottomneighbor window%
                                                                    09328
                      LOOP                                          09329
                         BEGIN                                      09330
                         (prcm13):                                  09331
                         [[di]].copcode _ clear;                    09332
                         cline(da.daleft+ls.rtx1,
                         da.datop+ls.rty,MIN( ls.rtx2-ls.rtx1
                         +ls.rthinc, da.daright-da.daleft-
                         ls.rtx1 ));  %clear only as far as the
```

```
                        RH edge of the window%                  09333
                        IF ls.rtlplink THEN                     09334
                           &ls _ &ls + ls.rtlplink              09335
                        ELSE EXIT LOOP;                         09336
                        END                                     09337
                     ELSE                                       09338
                        dline(da.daleft, ls.rty+da.datop);      09339
                     END;                                       09340
               %inserts%                                        09341
                  IF scwindow THEN                              09342
                     BEGIN                                      09343
                     wi _ wtop-1; %do scrolls instead of inserts%
                                                                09344
                     WHILE wi >= $ws DO                         09345
                        BEGIN %for each contiguous group of inserts%
                                                                09346
                        nline _ 0;                              09347
                        WHILE wi >= $ws DO                      09348
                           BEGIN                                09349
                           IF [[wi]].copcode = writelsg THEN    09350
                              BEGIN                             09351
                              &ls _ [[wi]].cparm1 + rttop;      09352
                              line _ ls.rty;                    09353
                              IF nline=0 OR line=tdline+nline THEN
                                                                09354
                                 BEGIN                          09355
                                 IF nline=0 THEN tdline _ line; 09356
                                 BUMP nline;                    09357
                                 END                            09358
                              ELSE EXIT;                        09359
                              END;                              09360
                           BUMP DOWN wi;                        09361
                           END;                                 09362
                        IF nline THEN doscroll(da.daleft, pdaright,
                        da.datop+tdline, pdabottom, -nline);    09363
                        END;                                    09364
                     END;                                       09365
                  FOR wi _ wtop-1 DOWN UNTIL < $ws DO           09366
                     IF [[wi]].copcode = writelsg THEN          09367
                        BEGIN                                   09368
                        &ls _ [[wi]].cparm1 + rttop;            09369
                        (prcm14):                               09370
                        IF NOT scwindow THEN                    09371
                           %scrolling hasn't been done, must do
                           inserts%                             09372
                           BEGIN                                09373
                           IF da.dalneighbor OR da.darneighbor OR
                           da.dabneighbor OR da.datneighbor THEN
                                                                09374
                              BEGIN                             09375
                              (prcm15):                         09376
                              [[wi]].copcode _ replace;         09377
                              &ls _ [[wi]].cparm1 + rttop;      09378
                              cline(da.daleft, da.datop+ls.rty,
                              da.daright-da.daleft);            09379
                              END                               09380
```

```
                    ELSE                                        09381
                       BEGIN                                    09382
                       (prcm16):                                09383
                       IF ls.rty + da.datop = lpymax THEN
                                                                09384
                          cline(da.daleft, ls.rty+da.datop,
                          da.daright-da.daleft)                 09385
                             %can't do insert on last line on
                             screen %                           09386
                       ELSE %insert new line%                   09387
                          inline(da.daleft, ls.rty+da.datop,
                          $"", FALSE);                          09388
                       IF ls.rtx1 > 0 AND (lptype .A 17B) =
                       deltadata THEN %must pad for display
                       delay%                                   09389
                          cline(da.daleft, ls.rty+da.datop,
                          ls.rtx1);                             09390
                       END;                                     09391
                    END;                                        09392
                 wrtstr(&da, &ls);                              09393
                 ls.rtlsid _ da.dahandle := da.dahandle+1;
                                                                09394
                 WHILE ls.rtlplink DO                           09395
                    BEGIN                                       09396
                    &ls _ &ls + ls.rtlplink;                    09397
                    ls.rtlsid _ da.dahandle := da.dahandle+1;
                                                                09398
                    END;                                        09399
                 END;                                           09400
%moves%                                                         09401
   %write them where they are supposed to be%                   09402
      FOR mi _ mtop-1 DOWN UNTIL < $ms DO                       09403
         IF [[mi]].copcode = movelsg THEN                       09404
            BEGIN                                               09405
            &ls _ [[mi]].cparm1 + rttop;                        09406
            (prcm21):                                           09407
            % do a cline where this line segment is
            supposed to be.%                                    09408
               cline(da.daleft, ls.rty+da.datop,
               (da.daright-da.daleft)/ls.rthinc);
                                                                09409
            %fix up byte pointers if necessary%                 09410
               %
               This plex commented out (percents
               doubled, too) because it seems to be
               responsible for messing up statement
               numbers with split screens                       09411
               &dls _ &ls;                                      09412
               LOOP                                             09413
                  BEGIN                                         09414
                  (prcm22):                                     09415
                  IF NOT dls.rtstid.stastr THEN %%must
                  get SDB%%                                     09416
                     BEGIN                                      09417
                     IF NOT lodprop( dls.rtstid,
                     txttyp : base) THEN                        09418
```

```
                                    err($"No text block associated
                                    with node");                09419
                                 base _ base + sdbhdl;          09420
                                    %%add in length of sdb header
                                    to get base of char string%%
                                                                09421
                                 end _ dls.rtbpe.bpadr -
                                 dls.rtbps.bpadr;               09422
                                    %%number of words between
                                    start and end%%             09423
                                 dls.rtbps _ stbptr(dls.rtccnt -
                                 1) + base;                     09424
                                 dls.rtbpe.bpadr _ dls.rtbps.bpadr
                                 + end;                         09425
                                 IF dls.rtlplink THEN &dls _ &dls
                                 + dls.rtlplink                 09426
                                 ELSE EXIT LOOP;                09427
                                 END;                           09428
                              END;                              09429
                           %                                    09430
                        IF (ls.rtbps.bpadr = ls.rtbpe.bpadr AND
                        ls.rtbps.bpbitpos < ls.rtbpe.bpbitpos)
                        THEN NULL %no string to be written%     09431
                        ELSE wrtstr(&da, &ls);                  09432
                        END;                                    09433
                  END;                                          09434
               ENDCASE; %no moves, writes, or deletes%          09435
            END; %display device%                               09436
         ENDCASE                                                09437
            BEGIN                                               09438
            UNTIL NOT cnxtaddress(&entry, curblk, backward  :&entry,
            curblk ) DO                                         09439
               BEGIN                                            09440
               CASE entry.copcode OF                            09441
                  =writelsg:                                    09442
                     BEGIN                                      09443
                     writend _ &entry; %last entry to process%  09444
                     UNTIL NOT cnxtaddress (&entry, curblk, backward
                     :&entry, curblk) OR entry.copcode # writelsg DO
                     NULL;                                      09445
                     %mark start of writes for later%           09446
                     savnxt _ &entry;                           09447
                     &savblk _ curblk;                          09448
                     %process writes in reverse order%          09449
                     DO                                         09450
                        BEGIN                                   09451
                        &ls _ entry.cparm1 + rttop;             09452
                        IF ls.rtnew AND NOT ls.rtlsid THEN      09453
                           wrtstr(&da, &ls)                     09454
                        END                                     09455
                     UNTIL NOT cnxtaddress (&entry, curblk, forward
                     :&entry, curblk) OR &entry = writend + 1;  09456
                     &entry _ savnxt;                           09457
                     curblk _ &savblk;                          09458
                     END;                                       09459
                  =movelsg:                                     09460
```

```
                    BEGIN                                                09461
                    &ls _ entry.cparm1 + rttop;                          09462
                    ls.rtbps _ -1; %old string%                          09463
                    ls.rtbpe _ 0;                                        09464
                    IF ls.rty = da.dabottom - da.datop THEN              09465
                       ls.rty _ ls.rty - 256;                            09466
                       %if a cord is zero, the strda jsys assumes you
                       do not want that cord changed--see wrtstr and
                       djsys documentation for strda%                    09467
                       wrtstr(&da, &ls);                                 09468
                       END;                                              09469
                 =deletlsg:                                              09470
                    BEGIN                                                09471
                    &ls _ entry.cparm1 + rttop;                          09472
                    ls.rtbps _ ls.rtbpe _ 0; %delete%                    09473
                    wrtstr(&da, &ls);                                    09474
                    END;                                                 09475
                 =blankda:                                               09476
                    BEGIN                                                09477
                    clearda(&da);                                        09478
                    END;                                                 09479
                 =suppressda:                                            09480
                    supda(&da);                                          09481
                 =restoreda:                                             09482
                    resda(&da);                                          09483
                 ENDCASE err($"System error:  undefined display
                 command...
                 ^C and RESET");                                         09484
              END; %of main command processing loop%                     09485
           END; %display device%                                         09486
     %deallocate all but first command storage block%                    09487
     cmdfree(dpcmblk);                                                   09488
     %free any outstanding signature or statement number strings - this
     code not currently in use - being freed in daupdate and clrall%
                                                                         09489
     %IF NOT da.davspec.vsindf AND NOT da.davspec.vsstnf THEN RETURN;
                                                                         09490
     end _ da.dalsz * lsrtl + da.dalsrt;                                 09491
     &ls _ da.dalsrt;                                                    09492
     DO                                                                  09493
        BEGIN                                                            09494
        IF NOT ls.rtexis THEN RETURN;                                    09495
        CASE ls.rtsrce OF                                                09496
           =srcsig, =srcstno:                                            09497
              <STGMGT, freestring>(ls.rtbps.bpadr, $dspblk);             09498
        ENDCASE NULL;                                                    09499
        END                                                              09500
     UNTIL (&ls _ &ls + lsrtl) >= end;%                                  09501
     RETURN END.                                                         09502
                                                                         09503
(wrtstr) PROCEDURE (da, ls); %write string in display area%              06641
     %write a string onto the display.  Uses the information in the
     display area and line segment records to determiine what to write
     and where and how.  If the rtbps field is -1 then the old string
     is used.%                                                           06642
     %-----------------%                                                06643
```

```
   LOCAL nls, store, savestart, save, startbp, endbp, i, char, l, mx,
   blanks, slength, bp, length, format, xposn, realrtx2;          06644
   LOCAL STRING send[200];                                        06645
   REF da, ls, nls;                                               06646
   CASE nldevice OF                                               06647
      = devlproc:                                                 06648
         BEGIN                                                    06649
         %send position cursor code%                              06650
            position( xposn _ ls.rtx1+da.daleft, ls.rty+da.datop );
                                                                  06651
         %send text%                                              06652
            LOOP                                                  06653
               BEGIN                                              06654
               startbp _ ls.rtbps;                                06655
               endbp _ ls.rtbpe;                                  06656
               IF ls.rtx2 = 777777B THEN realrtx2 _ -1            08476
               ELSE realrtx2 _ ls.rtx2; %for first char <ESC>, etc%
                                                                  08477
                  %rtx2 points to the last char of the string%    06658
               save _ MIN(da.daright, realrtx2+da.daleft);        06657
               savestart _ store _ chbmty + $send;                06659
               *send* _ NULL;                                     06660
               l _ send.L;  mx _ send.M;                          06661
               blanks _ 0;                                        06662
               i _ ls.rtx1+da.daleft;                             06663
               FOR i UP ls.rthinc UNTIL > save DO                 06664
                  BEGIN                                           06665
                  IF startbp = endbp THEN %fill with blanks%      06666
                     BUMP blanks                                  06667
                  ELSE                                            06668
                     BEGIN                                        06669
                     CASE (char _ ^startbp) OF                    06670
                        = TAB: char _ lptab;                      06671
                        = EOL: char _ CR;                         06672
                        ENDCASE;                                  06673
                     %*send* _ *send*, char;%                     06674
                        ^store _ char;                            06675
                        BUMP l;                                   06676
                     END;                                         06677
                  IF l >= mx THEN %time to sout -- string full%   06678
                     BEGIN                                        06679
                     !sout(dspjfn, savestart, -1);                06680
                     IF ldspjfn THEN                              06681
                        !sout(ldspjfn, savestart, -1);            06682
                     store _ savestart;                           06683
                     xposn _ xposn + l;                           07621
                     *send* _ NULL;                               06684
                     l _ send.L;                                  06685
                     END;                                         06686
                  END;                                            06687
               IF l THEN %sout the string%                        06688
                  BEGIN                                           06689
                  !sout(dspjfn, savestart, -1);                   06690
                  IF ldspjfn THEN                                 06691
                     !sout(ldspjfn, savestart, -1);               06692
                  store _ savestart;                              06693
```

```
                xposn _ xposn + 1;                                  07622
                *send* _ NULL;                                      06694
                l _ send.L;                                         06695
                END;                                                06696
             blanks _ MIN(blanks, da.daright -
             MIN(realrtx2,xposn));                                  07623
             IF blanks IN [1, lpxmax] THEN                          06697
                cline(current, current, blanks);                    06698
             %check to see if several line segments are to be
             writtten on this line%                                 06699
                IF ls.rtlplink THEN  %there is another line segment
                -- loop again%                                      06700
                   BEGIN                                            06701
                   &nls _ &ls + ls.rtlplink;                        06702
                   blanks _ nls.rtx1 - realrtx2 -1;                 06703
                   IF blanks IN [1, lpxmax] THEN                    06704
                      cline(current, current, blanks);              06705
                   &ls _ &nls;                                      06706
                   REPEAT LOOP;                                     06707
                   END                                              06708
                ELSE EXIT LOOP;                                     06709
             END;                                                   06710
          track();                                                  06711
          END;                                                      06712
       = imlac0, = imlac1:                                          06713
          BEGIN                                                     06714
          CASE ls.rtbps OF                                          06715
             = 0: %delete string%                                   06716
                BEGIN                                               06717
                IF ls.rtlsid = 0 OR NOT chkbit(da.dalsidb, ls.rtlsid,
                lsbtsize) THEN BEGIN                                06718
                   dismes(2, $"wrtstr: suppressing a non-existant
                   string");                                        06719
                   RETURN;                                          06720
                   END;                                             06721
                *send* _ begmsg, 5+remfudge, remssda,
                da.dahandle.daidr1 + remfudge,
                da.dahandle.daidr2 + remfudge, ls.rtlsid, TRUE;     06722
                !sout(dspjfn, chbmty + $send ,-send.L);             06723
                dassnbit(da.dalsidb,ls.rtlsid,lsbtsize);            06724
                RETURN;                                             06725
                END;                                                06726
             = -1: %use old string%                                 06727
                IF da.daseq THEN RETURN                             06728
                ELSE slength _ -1;                                  06729
             ENDCASE %new string%                                   06730
                BEGIN %get string length%                           06731
                startbp _ ls.rtbps;                                 06732
                endbp _ ls.rtbpe;                                   06733
                slength _slngth(startbp, endbp);                    06734
                END;                                                06735
          %setup format info and assign or check stid
          if default values, use default constants %                06736
          format _ 0;                                               06737
          save _ (da.dabottom-da.datop-ls.rty)/256;                 06738
          IF ls.rtlsid = 0 THEN  %new string%                       06739
```

```
    BEGIN                                                   06740
    ls.rtlsid _  assnbit(da.dalsidb,lsbtsize);              06741
    format.xyds _ FALSE;                                    06742
    END                                                     06743
ELSE                                                        06744
    BEGIN                                                   06745
    IF NOT chkbit(da.dalsidb,ls.rtlsid,lsbtsize) THEN       06746
       BEGIN                                                06747
       dismes(2,$"wrtstr: lsbitable fouled - CHKBIT
       failure!");                                          06748
       RETURN;                                              06749
       END;                                                 06750
    format.xyds _ IF ls.rtx1 OR save THEN FALSE ELSE TRUE;
                                                            06751
    END;                                                    06752
IF ls.rtfont = da.dafont THEN format.fdd _ TRUE             06753
ELSE format.fdd _ format.fds _ FALSE;                       06754
IF ls.rtcsize = da.dacsize THEN format.sdd _ TRUE           06755
ELSE format.sdd _ format.sds _ FALSE;                       06756
IF ls.rthinc = da.dahinc THEN format.idd _ TRUE             06757
ELSE format.idd _ format.ids _ FALSE;                       06758
%send command parameters plus string%                       06759
    length _ slength _                                      06760
       MIN((da.daright-ls.rtx1)/ls.rthinc, slength); %safety
       precausion%                                          06761
    IF da.daseq THEN % if sequential then use APPEND command%
                                                            06762
       BEGIN                                                06763
       *send* _ begmsg, length+3+remfudge, apsda,
         da.dahandle.daidr1+remfudge,
       da.dahandle.daidr2+remfudge;                         06764
       bp _ chbptr(send.L) + $send;                         06765
       END                                                  06766
    ELSE                                                    06767
       BEGIN                                                06768
       %compute message length%                             06769
       length _ MAX(0, length) + 6 + remfudge;              06770
          IF NOT format.xyds THEN length _ length + 4;      06771
          IF NOT format.sds AND NOT format.sdd THEN BUMP
          length;                                           06772
          IF NOT format.ids AND NOT format.idd THEN BUMP
          length;                                           06773
          IF NOT format.fds AND NOT format.fdd THEN BUMP
          length;                                           06774
       %string manipulation as per (JOURNAL, 11726,)%       06775
          *send* _ begmsg, length,  nwstrda,
           da.dahandle.daidr1+remfudge,
          da.dahandle.daidr2+remfudge, ls.rtlsid,
           slength + 1, format + remfudge;                  06776
          bp _ chbptr(send.L) + $send;                      06777
       %cordinates%                                         06778
          IF NOT format.xyds THEN                           06779
             BEGIN                                          06780
             ^bp _ remfudge + (ls.rtx1/256).xc1;            06781
             ^bp _ remfudge + (ls.rtx1/256).xc2;            06782
             ^bp _ remfudge + save.yc1;                     06783
```

```
                    ^bp _ remfudge + save.yc2;                          06784
                    send.L _ send.L + 4;                                06785
                    END;                                                06786
                %character size%                                        06787
                    IF NOT format.sds AND NOT format.sdd THEN           06788
                    BEGIN                                               06789
                    ^bp _ ls.rtcsize;                                   06790
                    BUMP send.L;                                        06791
                    END;                                                06792
                %horizontal increment%                                  06793
                    IF NOT format.ids AND NOT format.idd THEN           06794
                    BEGIN                                               06795
                    ^bp _ (ls.rthinc/128) + 1;                          06796
                    BUMP send.L;                                        06797
                    END;                                                06798
                %font%                                                  06799
                    IF NOT format.fds AND NOT format.fdd THEN           06800
                    BEGIN                                               06801
                    ^bp _ ls.rtfont*2 + 1;                              06802
                    BUMP send.L;                                        06803
                    END;                                                06804
                END;                                                    06805
            %now send header and string%                                06806
            IF slength > 0 THEN                                         06807
                UNTIL startbp = endbp                                   06808
                    OR startbp.bpadr > endbp.bpadr                      06809
                    OR (send.L _ send.L +1) >= send.M DO                06810
                    ^bp _ IF (char _ ^startbp) = EOL THEN CR ELSE char;
                                                                        06811
            !sout(dspjfn, chbmty + $send, -send.L) ;                    06812
        END;                                                            06813
    ENDCASE                                                             06814
        err($"No Tasker");                                              06815
RETURN;                                                                 06816
END.
                                                                        06817
(cmdfree) PROCEDURE (blkadress); %free command blocks%                  01772
    %starting at blkadress, free all display command blocks but the
    original one (which has no previous block address).%                01773
    LOCAL savblk, curblk;                                               01774
    REF savblk;                                                         01775
    &savblk _ blkadress; %original starting block--last used%           01776
    UNTIL NOT savblk.cbprev DO                                          01777
        BEGIN                                                           01778
        curblk _ savblk.cbprev;                                         01779
        freeblk(&savblk - bhl, $dspblk);                                01780
        &savblk _ curblk;                                               01781
        END;                                                            01782
        dpcmblk _ $commands; %reset global block pointer%               01783
        RETURN END.                                                     01784
                                                                        01785
(cnxtaddress) PROCEDURE (entry, block, direction);                      01786
    %updates entry and block in direction specified%                    01787
    REF block;                                                          01788
    CASE direction OF                                                   01789
        =forward:                                                       01790
```

```
            BEGIN                                                   01791
            BUMP entry;                                             01792
            IF entry <= block.cmnxt - 1 THEN RETURN(TRUE, entry,
            &block);                                                01793
            BUMP DOWN entry; %restore previous value%               01794
            IF NOT block.cbnxt %no new block%                       01795
               THEN RETURN(FALSE, entry, &block);                   01796
            entry _ (&block _ block.cbnxt) + cbhl;                  01797
            RETURN(TRUE, entry, &block);                            01798
            END;                                                    01799
         =backward:                                                 01800
            BEGIN                                                   01801
            BUMP DOWN entry;                                        01802
            IF entry >= &block + cbhl THEN RETURN(TRUE, entry, &block);
                                                                    01803
            BUMP entry; %restore previous value%                    01804
            IF NOT block.cbprev THEN RETURN(FALSE, entry, &block);  01805
            &block _ block.cbprev;                                  01806
            entry _ block.cmlst; %last used%                        01807
            RETURN(TRUE, entry, &block);                            01808
            END;                                                    01809
         ENDCASE err($"FATAL DISPLAY SYSTEM ERROR...
         Type ^C and RESET");                                       01810
         END.                                                       01811
                                                                    01812
   (cmdinit) PROCEDURE(block, da); %initialize display command block%
                                                                    01813
      REF block;                                                    01814
      dpcmblk _ &block; %global current block%                      01815
      block.cbprev _ block.cbnxt _ 0;                               01816
      block.cmlst _ &block + 64 - cbhl;                             01817
      block.cmnxt _ &block + cbhl;                                  01818
      block.cmda _ da;                                              01819
      RETURN END.                                                   01820
                                                                    01821
%...clearing da's, tables ....%                                     01822
   (clrall) PROCEDURE (da, free);  %reset LSRT entries for the da%  01823
      %clrall has three functions
      --zeros the LSRT entries for the given display
      --frees storage used for statement numbers and signatures
      --reallocates LSRT space if it's not being used               01824
      In general, if da=0 then these happen for all random display
      areas.  If free = TRUE, free strings.%                        01825
      %---------------%                                            01826
      LOCAL                                                         01827
         end,   %end of LSRT table plus one entry%                  01828
         index,                                                     01829
         straddr,  %signature or stmt # string address%             01830
         daend,                                                     01831
         used,  %number of LSRT words actually in use%              01832
         oldtable,                                                  01833
         oldsize,                                                   01834
         count,                                                     01835
         lines;                                                     01836
      REF da, index;                                                01837
      IF NOT &da THEN                                               01838
```

```
      daend _ (&da _ $dpyarea) + dacnt * dal                          01839
   ELSE                                                               01840
      daend _ &da + dal;                                              01841
   DO                                                                 01842
      IF da.daexis AND da.dahandle AND NOT da.daseq AND NOT
      da.daauxiliary THEN                                             01843
         BEGIN                                                        01844
         &index _ da.dalsrt - lsrtl;                                  01845
         end _ da.dalsz * lsrtl + da.dalsrt;                          01846
         %free storage associated with strings%                       01847
            UNTIL (&index _ &index + lsrtl) >= end OR NOT
            index.rtexis DO                                           01848
               BEGIN                                                  01849
               IF free AND (index.rtsrce = srcstno OR index.rtsrce =
               srcsig) THEN                                           01850
                  BEGIN                                               01851
                  straddr _ index.rtbps.bpadr;                        01852
                  <STGMGT, freestring> (straddr - 1, $dspblk);        01853
                  END;                                                01854
               %zero entries - very important%                        01855
                  FOR count _ 0 UP  UNTIL =  lsrtl DO                 01856
                  [&index + count] _ 0;                               01857
                  END;                                                01858
         used _ &index - da.dalsrt;                                   01859
         lines _ da.damrow/da.davinc + 2;%number of useable lines +
         1%                                                           01860
         IF (oldsize _ da.dalsz ) > lines AND used/lsrtl < lines THEN
                                                                      01861
            BEGIN                                                     01862
            IF (da.dashrinkcnt _ da.dashrinkcnt + 1) > shrink THEN
                                                                      01863
               BEGIN                                                  01864
               oldtable _ (da.dalsrt := 0);                           01865
               IF NOT maklsrt($dspblk, lines, &da) %no room in zone%
               OR NOT oldtable %system error% THEN                    01866
                  BEGIN                                               01867
                  dismes(2, $"Allocator out of display space");       01868
                  da.dalsrt _ oldtable; %restore%                     01869
                  END                                                 01870
               ELSE                                                   01871
                  BEGIN                                               01872
                  mvbfbf(oldtable, da.dalsrt, oldsize * lsrtl);       01873
                  freeblk(oldtable - bhl, $dspblk);                   01874
                  oldtable _ aulsrt;                                  01875
                  IF NOT (aulsrt _ getblk(lines * lsrtl, $dspblk))
                  THEN                                                01876
                     BEGIN                                            01877
                     aulsrt _ oldtable;%use old table for aux%        01878
                     dismes(2, $"Non-fatal Display error due to space
                     considerations.  Try viewspec 'f'.");            01879
                     END                                              01880
                  ELSE                                                01881
                     BEGIN                                            01882
                     freeblk(oldtable - bhl, $dspblk);                01883
                     aulsrt _ aulsrt + bhl; %start of useable space%
                                                                      01884
```

```
                    END;                                                01885
                  END;                                                  01886
                END                                                     01887
              ELSE da.dashrinkcnt _ 0;                                  01888
              END;                                                      01889
            da.dacrow _ da.daccol _ 0;                                  01890
            END                                                         01891
      UNTIL (&da _ &da + dal) >= daend;                                 01892
      RETURN;                                                           01893
      END.                                                              01894
                                                                        01895
%...special da functions...%                                           01896
   (clearda) PROCEDURE(da); %clear display area%                        08902
      %If &da is zero then zap the display image of all of the text
      display areas.  Otherwise, zap only the given text display area
      image.%                                                           08903
      %----------------%                                               08904
      LOCAL  end, line, ls, dls, all, dacount, tempda, length,
      cdabottom, cdaright;                                              08905
      LOCAL STRING send[100];                                           08906
      REF da, ls, dls, tempda, ltvsda;                                  08907
      IF NOT &da THEN                                                   08908
         BEGIN                                                          08909
         end _ (&da _ $dpyarea) + dacnt*dal;                            08910
         all _ TRUE;                                                    08911
         END                                                            08912
      ELSE                                                              08913
         BEGIN                                                          08914
         end _ &da + dal;                                               08915
         all _ FALSE;                                                   08916
         dacount _ 0;                                                   08917
         FOR &tempda _ $dpyarea UP dal UNTIL >= $dpyarea + dacnt*dal DO
                                                                        08918
            IF tempda.daexis AND NOT tempda.daauxiliary THEN BUMP
            dacount;                                                    08919
         END;                                                           08920
      CASE nldevice OF                                                  08921
         = devlproc:                                                    08922
            BEGIN                                                       08923
            IF NOT all AND scwindow AND da.daexis AND da.dahandle THEN
                                                                        08924
               BEGIN                                                    08925
               cdabottom _ da.dabottom;                                 08926
               IF cdabottom#lpymax THEN BUMP DOWN cdabottom;            08927
               cdaright _ da.daright;                                   09538
               IF cdaright#lpxmax THEN BUMP DOWN cdaright;              09539
               doscroll(da.daleft, cdaright, da.datop, cdabottom,
               (cdabottom-da.datop+1));                                 08928
               END;                                                     08929
            IF (all OR dacount = 1) AND NOT noclrall %save tty window%
            THEN                                                        08930
               BEGIN %clear the screen and rewrite command lines%       08931
               % check first if there is anything in text area to erase%
                                                                        08932
                  IF dacnt = 1 AND da.dacrow = 0 AND da.daccol = 0 THEN
                  RETURN;                                               08933
```

```
            cscreen();                                                  08934
            IF (lptype .A 4M) = deltadata THEN                          08935
               cline(0, ltvsda.datop, lpxmax);                          08936
            msgreset _ TRUE;                                            08937
            *send* _ *dnstr*;                                           08938
            dn($send);                                                  08939
            dsubsys($ssysname);                                         08940
            *savcfl* _ NULL;                                            08941
            cfldsp();                                                   08942
            IF arowon THEN                                              08943
               BEGIN                                                    08944
               arowon _ FALSE;                                          08945
               an();                                                    08946
               END                                                      08947
            ELSE                                                        08948
               BEGIN                                                    08949
               arowon _ TRUE;                                           08950
               af();                                                    08951
               END;                                                     08952
            vspsav _ vspsav[1] _ 0;                                     08953
            dspvsp(da.davspec, da.davspc2, 3);                          08954
            &ls _ da.dalsrt;                                            08955
            UNTIL NOT ls.rtexis DO                                      08956
               BEGIN                                                    08957
               ls.rtlsid _ 0;                                           08958
               &ls _ &ls + lsrtl;                                       08959
               END;                                                     08960
            DO da.dahandle _ 1                                          08961
            UNTIL (&da _ &da+dal) >= end;                               08962
            END                                                         08963
         ELSE                                                           08964
            DO                                                          08965
               IF da.daexis AND da.dahandle THEN                        08966
                  BEGIN %delete all of the lines%                       08967
                  &ls _ da.dalsrt;                                      08968
                  UNTIL NOT ls.rtexis DO                                08969
                     BEGIN                                              08970
                     line _ ls.rty;                                     08971
                     &dls _ &ls;                                        08972
                     LOOP                                               08973
                        BEGIN                                           08974
                        % NOTE:  Possible danger here if linking of
                        line segments skips one -- could fail to
                        delete it.  But may not be possible to have
                        non-adjacent linked lsrt's -- then it would
                        be safe. %                                      08975
                        dls.rtlsid _ 0;                                 08976
                        IF dls.rtlplink THEN &dls _ &dls +
                        dls.rtlplink                                    08977
                        ELSE EXIT LOOP;                                 08978
                        END;                                            08979
                     IF (length _ MIN ( (dls.rtx2-ls.rtx1)/da.dahinc
                     + 1 , (da.daright - da.daleft)/da.dahinc )) IN
                     [1, lpxmax] THEN                                   08980
                        cline(da.daleft+ls.rtx1, da.datop + line,
                        length);                                        08981
```

```
      all _ TRUE;                                                    06435
      END                                                            06436
   ELSE                                                              06437
      BEGIN                                                          06438
      end _ &da + dal;                                               06439
      all _ FALSE;                                                   06440
      dacount _ 0;                                                   06441
      FOR &tempda _ $dpyarea UP dal UNTIL >= $dpyarea + dacnt*dal DO
                                                                     06442
         IF tempda.daexis THEN BUMP dacount;                         06443
      END;                                                           06444
   CASE nldevice OF                                                  06445
      = devlproc:                                                    06446
         BEGIN                                                       06447
         IF all OR dacount = 1 THEN %clear the screen and rewrite
         command lines for single da %                               06448
            BEGIN                                                    06449
            cscreen();                                               06450
            IF (lptype .A 4M) = deltadata THEN                       06451
               cline(0, ltvsda.datop, lpxmax);                       06452
            dn($dnstr);                                              06453
            *savcfl* _ NULL;                                         06454
            cfldsp();                                                06455
            IF arowon THEN                                           06456
               BEGIN                                                 06457
               arowon _ FALSE;                                       06458
               an();                                                 06459
               END                                                   06460
            ELSE                                                     06461
               BEGIN                                                 06462
               arowon _ TRUE;                                        06463
               af();                                                 06464
               END;                                                  06465
            vspsav _ vspsav[1] _ 0;                                  06466
            dspvsp(da.davspec, da.davspc2, 3);                       06467
            &ls _ da.dalsrt;                                         06468
            UNTIL NOT ls.rtexis DO                                   06469
               BEGIN                                                 06470
               ls.rtlsid _ 0;                                        06471
               &ls _ &ls + lsrtl;                                    06472
               END;                                                  06473
            DO                                                       06474
               BEGIN                                                 06475
               da.dahandle _ 1;                                      06476
               da.dasuppress _ TRUE;                                 06477
               END                                                   06478
            UNTIL (&da _ &da+dal) >= end;                            06479
            END                                                      06480
         ELSE                                                        06481
            DO                                                       06482
               IF da.daexis AND da.dahandle THEN                     06483
                  BEGIN %delete all of the lines%                    06484
                  &ls _ da.dalsrt;                                   06485
                  UNTIL NOT ls.rtexis DO                             06486
                     BEGIN                                           06487
                     line _ ls.rty;                                  06488
```

```
                    &dls _ &ls;                                          06489
                    LOOP                                                 06490
                       BEGIN                                             06491
                       dls.rtlsid _ 0;                                   06492
                       IF dls.rtlplink THEN &dls _ &dls +
                       dls.rtlplink                                      06493
                       ELSE EXIT LOOP;                                   06494
                       END;                                              06495
                    IF (length _ (dls.rtx2-ls.rtx1)/da.dahinc + 1)
                    IN [1, lpxmax] THEN                                  06496
                       cline(da.daleft+ls.rtx1, da.datop + line,
                       length);                                          06497
                    &ls _ &dls + lsrtl;                                  06498
                    END;                                                 06499
                 da.dahandle _ 1;                                        06500
                 da.dasuppress _ TRUE;                                   06501
                 END                                                     06502
           UNTIL (&da _ &da+dal) >= end;                                 06503
        END;                                                             06504
     = imlac0, = imlac1:                                                 06505
        BEGIN                                                            06506
        DO                                                               06507
           IF da.daexis AND NOT da.daseq AND da.dahandle AND NOT
           da.daauxiliary THEN                                           06508
              BEGIN                                                      06509
              da.dasuppress _ TRUE;                                      06510
              *send* _ begmsg, 4+remfudge, remsdda,                      06511
                 da.dahandle.daidr1 + remfudge,                          06512
                 da.dahandle.daidr2 + remfudge,                          06513
                 FALSE;                                                  06514
              !sout(dspjfn, chbmty + $send ,-send.L);                    06515
              IF da.dalhandle THEN                                       06516
                 BEGIN                                                   06517
                 %suppress linked guy too%                               06518
                    *send* _ begmsg, 4+remfudge, remsdda,                06519
                       da.dalhandle.daidr1 + remfudge,                   06520
                       da.dalhandle.daidr2 + remfudge,                   06521
                       FALSE;                                            06522
                    !sout(ldspjfn, chbmty + $send ,-send.L);             06523
                 END;                                                    06524
              END                                                        06525
        UNTIL (&da _ &da+dal) >= end;                                    06526
        END;                                                             06527
     ENDCASE                                                             06528
        err($"No Tasker");                                               06529
  RETURN;                                                                06530
  END.
                                                                         06531
(resda) PROCEDURE(da); %restore display area%                            06382
  *If &da is zero then restore the display image of all of the text
  display areas.  Otherwise, restore only the given text display
  area image.%                                                           06383
  %----------------%                                                    06384
  LOCAL  end;                                                            06385
  LOCAL STRING send[200];                                                06386
  REF da;                                                                06387
```

```
   IF NOT &da THEN                                                    06388
      end _ (&da _ $dpyarea) + dacnt*dal                              06389
   ELSE                                                               06390
      end _ &da + dal;                                                06391
   CASE nldevice OF                                                   06392
      = devlproc:                                                     06393
         BEGIN                                                        06394
         DO                                                           06395
            IF da.daexis AND NOT da.daseq AND da.dahandle AND
            da.dasuppress AND NOT da.daauxiliary THEN                 06396
               BEGIN                                                  06397
               da.dasuppress _ FALSE;                                 06398
               %call the formatter%                                   06399
                  dafrmt(&da, 0);                                     06400
               END                                                    06401
         UNTIL (&da _ &da+dal) >= end;                                06402
         END;                                                         06403
      = imlac0, = imlac1:                                             06404
         BEGIN                                                        06405
         DO                                                           06406
            IF da.daexis AND NOT da.daseq AND da.dahandle AND
            da.dasuppress AND NOT da.daauxiliary THEN                 06407
               BEGIN                                                  06408
               %send command to restore it%                           06409
                  *send* _ begmsg, 3+remfudge, remrdda,
                  da.dahandle.daidr1 + remfudge,
                  da.dahandle.daidr2 + remfudge;                      06410
                  !sout(dspjfn, chbmty + $send ,-send.L);             06411
               IF da.dalhandle THEN                                   06412
                  BEGIN                                               06413
                  %restore linked guy too%                            06414
                     *send* _ begmsg, 3+remfudge, remrdda,
                     da.dalhandle.daidr1 + remfudge,
                     da.dalhandle.daidr2 + remfudge;                  06415
                     !sout(ldspjfn, chbmty + $send ,-send.L);         06416
                  END;                                                06417
               da.dasuppress _ FALSE;                                 06418
               END                                                    06419
         UNTIL (&da _ &da+dal) >= end;                                06420
         END;                                                         06421
      ENDCASE                                                         06422
         err ($"No Tasker");                                          06423
   RETURN;                                                            06424
   END.

                                                                      06425
(alocda) PROCEDURE (da); %allocate a display area%                    06322
   %uses the information in the display area record to allocate the
   display area.  The system display area handle is stored in the
   dahandle field.%                                                   06323
   %-----------------%                                               06324
   LOCAL save;                                                        06325
   LOCAL STRING send[20];                                             06326
   REF da;                                                            06327
   CASE nldevice OF                                                   06328
      = devlproc:                                                     06329
         BEGIN                                                        06330
```

```
            da.dahandle _ 1;                                                    06331
            da.dalhandle _ 0;                                                   06332
            END;                                                                06333
         = imlac0, = imlac1:                                                    06334
            BEGIN                                                               06335
            da.dahandle _ assnbit($dabt, dabtsize);                             06336
            IF da.dalsidb THEN                                                  06337
               dassnbit(da.dalsidb, -1, lsbtsize);                              06338
            *send* _ begmsg, 14+remfudge, nwada,                                06339
               %daid%                                                           06340
                  da.dahandle.daidr1 + remfudge,                                06341
                  da.dahandle.daidr2 + remfudge,                                06342
               %number of lines%                                                06343
                  (da.dabottom - da.datop)/da.davinc,                           06344
               %number of columns%                                              06345
                  (da.daright - da.daleft)/da.dahinc,                           06346
               %coords of lower left corner -- y must be reversed here
               (in NLS 0,0 is UPPER left, for display 0,0 is LOWER left.
               Also must divide by 256 for display resolution.)%                06347
                  (da.daleft/256).xc1 + remfudge,                               06348
                  (da.daleft/256).xc2 + remfudge,                               06349
                  (save _(bmarg-da.dabottom)/256).yc1 + remfudge,               06350
                  save.yc2 + remfudge,                                          06351
               %vinc%                                                           06352
                  (da.davinc/256).xc1 + remfudge,                               06353
                  (da.davinc/256).xc2 + remfudge,                               06354
               %format defaults%                                                06355
                  da.dacsize, da.dahinc/256, da.dafont;                         06356
            !sout(dspjfn, chbmty+$send, -send.L);                               06357
            END;                                                                06358
         ENDCASE                                                                06359
            err($"No Tasker");                                                  06360
      RETURN(da.dahandle);                                                      06361
      END.
                                                                                06362
   (dealocda) PROCEDURE (da);  %given a system daid, deallocate it%
                                                                                06363
      LOCAL STRING send[20];                                                    06364
      REF da;                                                                   06365
      CASE nldevice OF                                                          06366
         = devlproc: NULL;                                                      06367
         = imlac0, = imlac1: %send out string%                                  06368
            BEGIN                                                               06369
            *send* _ begmsg, 3+remfudge, remdda,                                06370
               da.dahandle.daidr1+remfudge,                                     06371
               da.dahandle.daidr2+remfudge;                                     06372
            !sout(dspjfn, chbmty+$send, -send.L);                               06373
            dassnbit($dabt, da.dahandle, dabtsize);                             06374
            da.dahandle _ 0;                                                    06375
            END;                                                                06376
         ENDCASE %local tasker%                                                 06377
            err ($"No Tasker");                                                 06378
      RETURN;                                                                   06379
      END.                                                                      06380
                                                                                06381
%...graphics primitives for alphanumeric (line-processor) displays %
```

```
                                                                    02296
(lpdspstr) PROCEDURE (x, y, string, standoutflag); %display the
specified string at specified location on a line processor
alpha-numeric display.  IF x = current then use current position of
cursor.  If standoutflag is TRUE then cause the string to standout%
                                                                    02297
   LOCAL store, start, char, l, mx;                                 02298
   LOCAL STRING send[200];                                          02299
   REF string;                                                      02300
   IF x NOT= current THEN position(x, y);                           02301
   IF standoutflag THEN standout();                                 02302
   CCPOS SF(*string*);                                              02303
   start _ store _ chbmty + $send;                                  02304
   *send* _ NULL;                                                   02305
   l _ send.L;  mx _ send.M;                                        02306
   LOOP                                                             02307
      BEGIN                                                         02308
      CASE char _ READC OF                                          02309
         = ENDCHR: EXIT LOOP;                                       02310
         = TAB:                                                     02311
            BEGIN                                                   02312
            ^store _ lptab; BUMP l;  %*send* _ *send*, lptab;%      02313
            END;                                                    02314
         ENDCASE                                                    02342
            BEGIN                                                   02343
            ^store _ char; BUMP l; %*send* _ *send*, char;%         02344
            END;                                                    02345
      IF l >= mx THEN %time to sout%                                02346
         BEGIN                                                      02347
         !sout(dspjfn, start, -1);                                  02348
         IF ldspjfn THEN                                            03314
            !sout(ldspjfn, start, -1);                              03315
         *send* _ NULL;                                             02349
         l _ send.L;                                                02350
         store _ start;                                             02351
         END;                                                       02352
      END;                                                          02353
   IF l THEN %output it%                                            02354
      BEGIN                                                         03317
      !sout(dspjfn, start, -1);                                     02355
      IF ldspjfn THEN                                               03316
         !sout(ldspjfn, start, -1);                                 03319
      END;                                                          03318
   IF standoutflag THEN endstandout();                              02356
   track();                                                         02357
   RETURN;                                                          02358
   END.                                                             02359

                                                                    02360
(pad) PROCEDURE (count);  %pad with count null characters%          02361
   %adjusts for current line speed by dividing count by lpbaudfactor%
                                                                    02362
   count _ MIN((count/lpbaudfactor) + 1, padstr.M);                 02363
   IF count > 0 THEN                                                02364
      BEGIN                                                         03320
      !sout(dspjfn, chbmty + $padstr, -count);                      02365
```

```
      IF ldspjfn THEN                                                  03322
         !sout(ldspjfn, chbmty + $padstr, -count);                     03323
      END;                                                             03321
   RETURN;                                                             02366
   END.                                                                02367

                                                                       02368
(position) PROCEDURE % position cursor - alphanumeric displays %
                                                                       08554
   ( xpos,  % x position %                                             08555
     ypos); % y position %                                             08556
   LOCAL STRING send[10];  %for sout%                                  08557
   IF xpos NOT IN [0, lpxmax] OR ypos NOT IN [0, lpymax] THEN          08558
      err($"Illegal coordinate detected in POSITION");                 08559
   ypos _ lpymax-ypos;                                                 08560
   *send* _ lpesc, lpposition;                                         08561
   apcoord(xpos, $send);                                               08562
   apcoord(ypos, $send);                                               08563
   tracking _ FALSE;                                                   08564
   !sout(dspjfn, chbmty + $send, -send.L);                             08565
   IF ldspjfn THEN                                                     08566
      !sout(ldspjfn, chbmty + $send, -send.L);                         08567
   RETURN;                                                             08568
   END.                                                                08569

                                                                       08570
(lpttywindow) PROCEDURE % specify tty window - alphanumeric displays
%                                                                      02379
   ( top,    % top line of window %                                    02380
     bottom  % bottom line of window %                                 02381
   );                                                                  02382
   LOCAL save;                                                         02775
   LOCAL STRING send[10];                                              02383
   IF top NOT IN [0, lpymax] OR bottom NOT IN [0, lpymax] THEN         02697
      err($"Illegal coordinate detected in LPTTYWINDOW");              02698
   *send* _ lpesc, lptty, lpymax-top+40B, lpymax-bottom+40B;           02384
   save _ tracking := FALSE; %to avoid printer output%                 02773
   !sout(dspjfn, chbmty + $send, -send.L);                             02385
   IF ldspjfn THEN                                                     03326
      !sout(ldspjfn, chbmty + $send, -send.L);                         03327
   tracking _ save; %to avoid printer output%                          02774
   RETURN;                                                             02386
   END.                                                                02387

                                                                       02388
(lprset) PROCEDURE; % reset alphanumeric displays %                    02389
   LOCAL save;                                                         02778
   LOCAL STRING send[10];                                              02390
   *send* _ lpesc, lpreset;                                            02391
   save _ tracking := FALSE; %to avoid printer output%                 02776
   !sout (dspjfn, chbmty + $send, -send.L);                            02392
   IF ldspjfn THEN                                                     03328
      !sout (ldspjfn, chbmty + $send, -send.L);                        03329
   pad(lpdlpad);                                                       02393
   cscreen(); % be sure screen cleared %                               02394
   tracking _ save; %to avoid printer output%                          02777
```

```
   RETURN;                                                          02395
   END.                                                             02396

                                                                    02397
(cscreen) PROCEDURE; % clear screen - alphanumeric displays %       02398
   LOCAL save;                                                      02781
   LOCAL STRING send[10];                                           02399
   %pause if timing (dismes) currently being done%                  08478
      timerpause();                                                 08479
   *send* _ lpesc, lpcscreen;                                       02400
   save _ tracking := FALSE; %to avoid printer output%              02779
   !sout(dspjfn, chbmty + $send, -send.L);                          02401
   IF ldspjfn THEN                                                  03330
      !sout(ldspjfn, chbmty + $send, -send.L);                      03331
   pad(lpdlpad);                                                    02402
   tracking _ save; %to avoid printer output%                       02780
   RETURN;                                                          02403
   END.                                                             02404

                                                                    02405
(track) PROCEDURE; % resume tracking - alphanumeric displays %      02406
   LOCAL STRING send[10];                                           02407
   *send* _ lpesc, lptrack;                                         02408
   !sout(dspjfn, chbmty + $send, -send.L);                          02409
   IF ldspjfn THEN                                                  03332
      !sout(ldspjfn, chbmty + $send, -send.L);                      03333
   tracking _ TRUE;                                                 02410
   IF tracksend THEN lppsr( tracksend := 0 );                       02749
   RETURN;                                                          02411
   END.                                                             02412

                                                                    02413
(cline) PROCEDURE % write blanks - alphanumeric displays %          08523
   (x,       %x coordinate or CURRENT%                              08524
   y,        %y coordinate or CURRENT%                              08525
   nchar );  % number of blanks to write %                          08526
   LOCAL STRING send[10];                                           08527
   IF nchar = 0 THEN RETURN;  %clear 0 positions%                   08528
   IF nchar > lpxmax THEN nchar _ lpxmax;  %sloppy code, but it gets
   rid of an error msg in non-dangerous situations%                 08529
   IF nchar NOT IN [1,lpxmax] THEN err($"Illegal number of blanks
   requested in CLINE");                                            08530
   IF x NOT= current THEN position(x, y);                           08531
   *send* _ lpesc, lpcline;                                         08532
   apcoord(nchar, $send);                                           08533
   !sout(dspjfn, chbmty + $send, -send.L);                          08534
   IF ldspjfn THEN                                                  08535
      !sout(ldspjfn, chbmty + $send, -send.L);                      08536
   pad(nchar);                                                      08537
   IF x NOT= current THEN track();                                  08538
   RETURN;                                                          08539
   END.                                                             08540

                                                                    08541
(apcoord)PROCEDURE(coord, str);                                     08542
   LOCAL coorhi;                                                    08543
```

```
   REF str;                                                          08544
   IF coord > 94 THEN                                                08545
      BEGIN                                                          08546
      coorhi _ coord.xc1 + 40B;                                      08547
      *str* _ *str*, cooresc, coorhi;                                08548
      coord _ coord.xc2;                                             08549
      END;                                                           08550
   *str* _ *str*, coord+40B;                                         08551
   RETURN;                                                           08552
   END.                                                              08553
(dline) PROCEDURE % delete line - alphanumeric displays %            02427
   (x,      %x coordinate or CURRENT%                                02428
   y);      %y coordinate or CURRENT%                                02429
   LOCAL STRING send[10];                                            02430
   IF x NOT= current THEN position(x, y);                            02431
   *send* _ lpesc, lpdline;                                          02432
   !sout(dspjfn, chbmty + $send, -send.L);                           02433
   IF ldspjfn THEN                                                   03336
      !sout(ldspjfn, chbmty + $send, -send.L);                       03337
   pad(lpdlpad);                                                     02434
   track();                                                          02435
   RETURN;                                                           02436
   END.                                                              02437

                                                                     02438
(inline) PROCEDURE % insert line - alphanumeric displays %           06305
   (x,              %x coordinate or CURRENT%                        06306
   y,               %y coordinate or CURRENT%                        06307
   string,          %string to be displayed%                         06308
   standoutflag); %FLAG -- TRUE: make the string standout%           06309
   LOCAL STRING send[10];                                            06310
   REF string;                                                       06311
   IF x NOT= current THEN position(x, y);                            06312
   *send* _ lpesc, lpinline;                                         06313
   !sout(dspjfn, chbmty + $send, -send.L);                           06314
   IF ldspjfn THEN                                                   06315
      !sout(ldspjfn, chbmty + $send, -send.L);                       06316
   IF lpilpad > 0 THEN pad(lpilpad);                                 06317
   lpdspstr( current, current, &string, standoutflag);               06318
   RETURN;                                                           06319
   END.                                                              06320

                                                                     06321
(doscroll) PROCEDURE % scroll window up or down - alphanumeric
displays %                                                           09504
   ( xleft,  % positions of boudaries %                              09505
     xright,                                                         09506
     ytop,                                                           09507
     ybot,                                                           09508
     nlines); % >0: up, <0: down %                                   09509
   LOCAL pddl, nlsign;                                               09510
   LOCAL STRING send[20];  %for sout%                                09511
   nlsign _ 1;                                                       09512
   IF nlines < 0 THEN                                                09513
      BEGIN                                                          09514
      nlsign _ -1;                                                   09515
```

```
      nlines _ - nlines;                                                09516
      END;                                                              09517
   nlines _ MIN(nlines, ybot-ytop+1);                                   09518
   nlines _ nlines*nlsign;                                              09519
   IF xleft NOT IN [0, lpxmax] OR xright NOT IN [0, lpxmax] OR ytop
   NOT IN [0, lpymax] OR ybot NOT IN [0, lpymax] THEN                   09520
      err($"Illegal coordinate detected in DOSCROLL");                  09521
   *send* _ lpesc, lpscroll;                                            09522
   apcoord(xleft, $send);                                               09523
   apcoord(xright, $send);                                              09524
   apcoord(lpymax-ytop, $send);                                         09525
   apcoord(lpymax-ybot, $send);                                         09526
   apcoord(nlines .A 777777B, $send);                                   09527
   !sout(dspjfn, chbmty + $send, -send.L);                              09528
   IF ldspjfn THEN                                                      09529
      !sout(ldspjfn, chbmty + $send, -send.L);                          09530
   pddl _ (xright-xleft)/25 + 2;                                        09531
   pad(pddl * (ybot-ytop+1));                                           09532
   RETURN;                                                              09533
   END.                                                                 09534

                                                                        09535
(lpmarkit) PROCEDURE % bug selection mark - alphanumeric displays %
                                                                        08571
   ( xpos,   % x position of char to bug %                              08572
     ypos); % y position of char to bug %                               08573
   LOCAL save;                                                          08574
   LOCAL STRING send[10];                                               08575
   IF xpos NOT IN [0, lpxmax] OR ypos NOT IN [0, lpymax] THEN           08576
      err($"Illegal coordinate detected in POSITION");                  08577
   ypos _ lpymax-ypos;                                                  08578
   *send* _ lpesc, lpbug;                                               08579
   apcoord(xpos, $send);                                                08580
   apcoord(ypos, $send);                                                08581
   save _ tracking := FALSE; %to avoid printer output%                  08582
   !sout(dspjfn, chbmty + $send, -send.L);                              08583
   IF ldspjfn THEN                                                      08584
      !sout(ldspjfn, chbmty + $send, -send.L);                          08585
   pad(8); %change to use a symbolic value!!!%                          08586
   tracking _ save; %to avoid printer output%                           08587
   RETURN;                                                              08588
   END.                                                                 08589

                                                                        08590
(lppopmark) PROCEDURE; % pop bug selection mark - alphanumeric
displays %                                                              02464
   LOCAL save;                                                          02785
   LOCAL STRING send[10];                                               02465
   *send* _ lpesc, lppbug;                                              02466
   save _ tracking := FALSE; %to avoid printer output%                  02786
   !sout(dspjfn, chbmty + $send, -send.L);                              02467
   IF ldspjfn THEN                                                      03343
      !sout(ldspjfn, chbmty + $send, -send.L);                          03344
   pad(8); %change to use a symbolic value!!!%                          02468
   tracking _ save; %to avoid printer output%                           02787
   RETURN;                                                              02469
```

```
   END.                                                                 02470

                                                                        02471
(intter) PROCEDURE; % interrogate alphanumeric displays %               02472
   LOCAL save;                                                          02790
   LOCAL STRING send[10];                                               02473
   *send* _ lpesc, lpintter;                                            02474
   save _ tracking := FALSE; %to avoid printer output%                  02789
   !sout(dspjfn, chbmty + $send, -send.L);                              02475
   IF ldspjfn THEN                                                      03345
      !sout(ldspjfn, chbmty + $send, -send.L);                          03346
   tracking _ save; %to avoid printer output%                           02788
   RETURN;                                                              02476
   END.                                                                 02477

                                                                        02478
(lpcmode) PROCEDURE; % set Lineprocessor to coordinate mode %           02479
   LOCAL save;                                                          02793
   LOCAL STRING send[10];                                               02480
   *send* _ lpesc, lpcoord;                                             02481
   save _ tracking := FALSE; %to avoid printer output%                  02792
   !sout(dspjfn, chbmty + $send, -send.L);                              02482
   IF ldspjfn THEN                                                      03347
      !sout(ldspjfn, chbmty + $send, -send.L);                          03348
   tracking _ save; %to avoid printer output%                           02791
   RETURN;                                                              02483
   END.                                                                 02484

                                                                        02485
(standout) PROCEDURE; % Send stand out following characters command -
alphanumeric displays %                                                 02486
   LOCAL STRING send[10];                                               02487
   *send* _ lpesc, lpstandout;                                          02488
   !sout(dspjfn, chbmty + $send, -send.L);                              02489
   IF ldspjfn THEN                                                      03349
      !sout(ldspjfn, chbmty + $send, -send.L);                          03350
   RETURN;                                                              02490
   END.                                                                 02491

                                                                        02492
(endstandout) PROCEDURE; % Send end stand out command - alphanumeric
displays %                                                              02493
   LOCAL STRING send[10];                                               02494
   *send* _ lpesc, lpendstandout;                                       02495
   !sout(dspjfn, chbmty + $send, -send.L);                              02496
   IF ldspjfn THEN                                                      03351
      !sout(ldspjfn, chbmty + $send, -send.L);                          03352
   RETURN;                                                              02497
   END.                                                                 02498

                                                                        02499
(lppopen) PROCEDURE; % open Lineprocessor printer %                     02535
   LOCAL STRING send[10];                                               02536
   *send* _ lpesc, lptptopen;                                           02537
   !sout(dspjfn, chbmty + $send, -send.L);                              02538
   %dont do shared screen sout!%                                        03357
```

```
    RETURN;                                                          02539
    END.                                                             02540

                                                                     02541
(lppnoper) PROCEDURE(reopenflag); %new open lp printer%              08504
    LOCAL STRING send[10];                                           08505
    IF newcp THEN                                                    08506
        BEGIN                                                        08507
        !gjinf();                                                    08508
        lppldev _ 40B + (r3 .A 77B);                                 08509
        *send* _ lpesc, lptnopn, lppldev, 40B+reopenflag;            08510
        !sout(dspjfn, chbmty + $send, -send.L);                      08511
        END                                                          08512
    ELSE                                                             08513
        BEGIN                                                        08514
        *send* _ lpesc, lptptopen;                                   08515
        !sout(dspjfn, chbmty + $send, -send.L);                      08516
        %dont do shared screen sout!%                                08517
        END;                                                         08518
    RETURN;                                                          08519
    END.                                                             08520
                                                                     08521
(lppclose) PROCEDURE; % close Lineprocessor printer %                02542
    LOCAL STRING send[10];                                           02543
    *send* _ lpesc, lptptclose;                                      02544
    !sout(dspjfn, chbmty + $send, -send.L);                          02545
    RETURN;                                                          02546
    END.                                                             02547

                                                                     02548
(lppsr) PROCEDURE % Lineprocessor printer string handler %           02549
    (lppcnt); % number of characters to send %                       02550
    % sends a string of lppcnt chars to Line Processor printer.  Gets
    them from procedure lppch %                                      02551
    LOCAL STRING send[200];                                          02552
    LOCAL store, l;                                                  02553
    IF &lppch=0 THEN err($"no LP character routine - lppsr");        02554
    IF lppcnt<=0 OR lppjfn=0 THEN RETURN;                            02555
    *send* _ NULL;                                                   02556
    l _ send.L;                                                      02557
    store _ chbmty + $send;                                          02558
    ^store _ lpesc;                                                  02559
    ^store _ lptptsr;                                                02560
    ^store _ 40B;                                                    02561
    ^store _ lppcnt+40B;                                             02562
    l _ l + 4;                                                       02563
    WHILE (lppcnt _ lppcnt-1) >= 0 DO                                02564
        BEGIN                                                        02565
        ^store _ lppch(); % get character and store it %             02566
        BUMP l;                                                      02567
        END;                                                         02568
    !sout(dspjfn, chbmty+$send, l);                                  02569
    !gtsts(lppjfn);                                                  02570
    IF r2 .A 1B9 THEN % end of file! %                               02571
        lppterm();                                                   02572
    RETURN;                                                          02573
```

```
    END.                                                                02574
(lppterm) PROCEDURE; % terminate LP printing %                          02575
    % aborts Line Processor printer operation %                        02576
    !dti(0); % deassign null %                                          02746
    !dic(4B5,200B); % deactivate chan 28 %                              02748
    lppclose();                                                         02577
    IF NOT SKIP !closf(lppjfn) THEN err($"cant close text file");
                                                                        02578
    lppjfn _ tracksend _ 0;                                             02579
    RETURN;                                                             02580
    END.                                                                02581
                                                                        02744
(lppstart) PROCEDURE % start LP printing from jfn %                     02729
    (jfn, % jfn for text file %                                         02730
    proc); % address of procedure to get chars from %                   02731
    % call this procedure to start printing on LP printer %            02732
    % setup variables %                                                02737
        lppjfn _ jfn;                                                   02733
        &lppch _ proc;                                                  02734
        tracksend _ lppcol _ lpplin _ 0;                                02735
    % setup PSI stuff for string requests from LP %                     02736
        chntab[28] _ 1B6+$lppint; % use channel 28 for interrupt %
                                                                        02738
        !ati(0+28); % designate null (zero) as interrupt char %         02739
        !aic(4B5,200B); % activate chn 28 %                             02740
    lppopen(); % send open string %                                     02741
    RETURN;                                                             02742
    END.                                                                02743
                                                                        02745
(lppint) PROCEDURE; % LP printer interupt routine%                      02751
    %---------------------%                                            02752
                                                                        02753
    %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%                  02754
                                                                        02755
    %---------------------%                                            02756
    %save registers%                                                    02757
    svac1 _ r1;                                                         02758
    r1 _ $svacs;                                                        02759
    !BLT r1, svacse;                                                    02760
    !MOVE r1,40B; sav40 _ r1;                                           02769
    s _ -100; !HRL s,s; !HRRI s,psistk;                                 02771
    m _ s;                                                              02772
    IF tracking THEN lppsr(16)                                          02761
    ELSE tracksend _ tracksend + 16;                                    02762
    %restore registers%                                                 02763
    r1 _ sav40; !MOVEM r1,40B;                                          02770
    !HRLZI r1, svacs;                                                   02764
    !BLT r1, 17B;                                                       02765
    r1 _ svac1;                                                         02766
    !debrk;                                                             02767
    END.                                                                02768
FINISH of DISPLAY                                                       02500
(wrtstr) PROCEDURE (da, ls); %write string in display area%             03116
    %write a string onto the display.  Uses the information in the
    display area and line segment records to determiine what to write and
```

```
where and how.  If the rtbps field is -1 then the old string is
used.%                                                          03117
%-----------------%                                             03118
LOCAL nls, store, savestart, save, startbp, endbp, i, char, l, mx,
blanks, slength, bp, length, format;                            03119
LOCAL STRING send[200];                                         03120
REF da, ls, nls;                                                03121
CASE nldevice OF                                                03122
   = devlproc:                                                  03123
      BEGIN                                                     03124
      %send position cursor code%                               03125
         position( i _ ls.rtx1+da.daleft, ls.rty+da.datop );    03126
      %send text%                                               03127
         LOOP                                                   03128
            BEGIN                                               03129
            startbp _ ls.rtbps;                                 03130
            endbp _ ls.rtbpe;                                   03131
            save _ MIN(da.daright, ls.rtx2+da.daleft);          03132
               %rtx2 points to the last char of the string%     03133
            savestart _ store _ chbmty + $send;                 03134
            *send* _ NULL;                                      03135
            l _ send.L;  mx _ send.M;                           03136
            blanks _ 0;                                         03137
            FOR i UP ls.rthinc UNTIL > save DO                  03138
               BEGIN                                            03139
               IF startbp = endbp THEN %fill with blanks%       03140
                  BUMP blanks                                   03141
               ELSE                                             03142
                  BEGIN                                         03143
                  CASE (char _ ^startbp) OF                     03144
                     = TAB: char _ lptab;                       03145
                     = EOL: char _ CR;                          03146
                     ENDCASE;                                   03147
                  %*send* _ *send*, char;%                      03148
                     ^store _ char;                             03149
                     BUMP l;                                    03150
                  END;                                          03151
               IF l >= mx THEN %time to sout -- string full%    03152
                  BEGIN                                         03153
                  !sout(dspjfn, savestart, -1);                 03154
                  store _ savestart;                            03155
                  *send* _ NULL;                                03156
                  l _ send.L;                                   03157
                  END;                                          03158
               END;                                             03159
            IF l THEN %sout the string%                         03160
               BEGIN                                            03161
               !sout(dspjfn, savestart, -1);                    03162
               store _ savestart;                               03163
               *send* _ NULL;                                   03164
               l _ send.L;                                      03165
               END;                                             03166
            IF blanks IN [1, lpxmax] THEN cline(current, current,
            blanks);                                            03167
            %check to see if several line segments are to be writtten
            on this line%                                       03168
```

```
            IF ls.rtlplink THEN  %there is another line segment --
            loop again%                                              03169
               BEGIN                                                 03170
               &nls _ &ls + ls.rtlplink;                             03171
               blanks _ nls.rtx1 - ls.rtx2 -1;                       03172
               IF blanks IN [1, lpxmax] THEN cline(current,
               current, blanks);                                     03173
               &ls _ &nls;                                           03174
               REPEAT LOOP;                                          03175
               END                                                   03176
            ELSE EXIT LOOP;                                          03177
         END;                                                        03178
      track();                                                       03179
      END;                                                           03180
   = imlac0, = imlac1:                                               03181
      BEGIN                                                          03182
      CASE ls.rtbps OF                                               03183
         = 0: %delete string%                                        03184
            BEGIN                                                    03185
            IF ls.rtlsid = 0 OR NOT chkbit(da.dalsidb, ls.rtlsid,
            lsbtsize) THEN BEGIN                                     03186
               dismes(2, $"wrtstr: suppressing a non-existant
               string");                                             03187
               RETURN;                                               03188
               END;                                                  03189
            *send* _ begmsg, 5+remfudge, remssda,
            da.dahandle.daidr1 + remfudge,
            da.dahandle.daidr2 + remfudge, ls.rtlsid, TRUE;          03190
            !sout(dspjfn, chbmty + $send ,-send.L);                  03191
            dassnbit(da.dalsidb,ls.rtlsid,lsbtsize);                 03192
            RETURN;                                                  03193
            END;                                                     03194
         = -1: %use old string%                                      03195
            IF da.daseq THEN RETURN                                  03196
            ELSE slength _ -1;                                       03197
         ENDCASE %new string%                                        03198
            BEGIN %get string length%                                03199
            startbp _ ls.rtbps;                                      03200
            endbp _ ls.rtbpe;                                        03201
            slength _slngth(startbp, endbp);                         03202
            END;                                                     03203
      %setup format info and assign or check stid
      if default values, use default constants %                     03204
      format _ 0;                                                    03205
      save _ (da.dabottom-da.datop-ls.rty)/256;                      03206
      IF ls.rtlsid = 0 THEN  %new string%                            03207
         BEGIN                                                       03208
         ls.rtlsid _  assnbit(da.dalsidb,lsbtsize);                  03209
         format.xyds _ FALSE;                                        03210
         END                                                         03211
      ELSE                                                           03212
         BEGIN                                                       03213
         IF NOT chkbit(da.dalsidb,ls.rtlsid,lsbtsize) THEN           03214
            BEGIN                                                    03215
            dismes(2,$"wrtstr: lsbitable fouled - CHKBIT failure!");
                                                                     03216
```

```
        RETURN;                                                 03217
        END;                                                    03218
      format.xyds _ IF ls.rtx1 OR save THEN FALSE ELSE TRUE;    03219
      END;                                                      03220
    IF ls.rtfont = da.dafont THEN format.fdd _ TRUE             03221
    ELSE format.fdd _ format.fds _ FALSE;                       03222
    IF ls.rtcsize = da.dacsize THEN format.sdd _ TRUE           03223
    ELSE format.sdd _ format.sds _ FALSE;                       03224
    IF ls.rthinc = da.dahinc THEN format.idd _ TRUE             03225
    ELSE format.idd _ format.ids _ FALSE;                       03226
    %send command parameters plus string%                       03227
      length _ slength _                                        03228
        MIN((da.daright-ls.rtx1)/ls.rthinc, slength); %safety
        precausion%                                             03229
      IF da.daseq THEN % if sequential then use APPEND command%
                                                                03230
        BEGIN                                                   03231
        *send* _ begmsg, length+3+remfudge, apsda,
          da.dahandle.daidr1+remfudge,
        da.dahandle.daidr2+remfudge;                            03232
        bp _ chbptr(send.L) + $send;                            03233
        END                                                     03234
      ELSE                                                      03235
        BEGIN                                                   03236
        %compute message length%                                03237
        length _ MAX(0, length) + 6 + remfudge;                 03238
          IF NOT format.xyds THEN length _ length + 4;          03239
          IF NOT format.sds AND NOT format.sdd THEN BUMP length;
                                                                03240
          IF NOT format.ids AND NOT format.idd THEN BUMP length;
                                                                03241
          IF NOT format.fds AND NOT format.fdd THEN BUMP length;
                                                                03242
        %string manipulation as per (JOURNAL, 11726,)%          03243
          *send* _ begmsg, length,  nwstrda,
           da.dahandle.daidr1+remfudge,
          da.dahandle.daidr2+remfudge, ls.rtlsid,
           slength + 1, format + remfudge;                      03244
          bp _ chbptr(send.L) + $send;                          03245
        %cordinates%                                            03246
          IF NOT format.xyds THEN                               03247
            BEGIN                                               03248
            ^bp _ remfudge + (ls.rtx1/256).xc1;                 03249
            ^bp _ remfudge + (ls.rtx1/256).xc2;                 03250
            ^bp _ remfudge + save.yc1;                          03251
            ^bp _ remfudge + save.yc2;                          03252
            send.L _ send.L + 4;                                03253
            END;                                                03254
        %character size%                                        03255
          IF NOT format.sds AND NOT format.sdd THEN             03256
            BEGIN                                               03257
            ^bp _ ls.rtcsize;                                   03258
            BUMP send.L;                                        03259
            END;                                                03260
        %horizontal increment%                                  03261
          IF NOT format.ids AND NOT format.idd THEN             03262
```

```
                    BEGIN                                                   03263
                    ^bp _ (ls.rthinc/128) + 1;                              03264
                    BUMP send.L;                                            03265
                    END;                                                    03266
                %font%                                                      03267
                   IF NOT format.fds AND NOT format.fdd THEN                03268
                    BEGIN                                                   03269
                    ^bp _ ls.rtfont*2 + 1;                                  03270
                    BUMP send.L;                                            03271
                    END;                                                    03272
                END;                                                        03273
             %now send header and string%                                   03274
             IF slength > 0 THEN                                            03275
                UNTIL startbp = endbp                                       03276
                   OR startbp.bpadr > endbp.bpadr                           03277
                   OR (send.L _ send.L +1) >= send.M DO                     03278
                   ^bp _ IF (char _ ^startbp) = EOL THEN CR ELSE char;
                                                                            03279
             !sout(dspjfn, chbmty + $send, -send.L) ;                       03280
          END;                                                              03281
       ENDCASE                                                              03282
          BEGIN                                                             03283
          r1.sdaid _ da.dahandle;                                           03284
          r1.sstrid _ ls.rtlsid;                                            03285
          save _ r1;                                                        03286
          r2 _ ls.rtbps;                                                    03287
          r3 _ ls.rtbpe;                                                    03288
          r4.sxcord _ ls.rtx1/256;                                          03289
          r4.sycord _ ((da.dabottom-da.datop) - ls.rty)/256;                03290
          %if default values, use default constants%                        03291
          IF ls.rtfont = da.dafont THEN r4.sfont _ -1                       03292
          ELSE r4.sfont _ ls.rtfont*2 + 1;                                  03293
          IF ls.rtcsize = da.dacsize THEN r4.scsize _ -1                    03294
          ELSE r4.scsize _ ls.rtcsize*2 + 1;                                03295
          IF ls.rthinc = da.dacsize THEN r4.shinc _ -1                      03296
          ELSE r4.shinc _ (ls.rthinc/128) + 1;                              03297
          IF NOT SKIP !JSYS strda THEN                                      03298
             IF r1 = syserr THEN NULL                                       03299
             ELSE                                                           03300
                dismes(2,$"NLS Display Error")                              03301
          ELSE ls.rtlsid _ r1 .A 18M;                                       03302
          IF da.dalhandle THEN %linked display area%                        03303
             BEGIN                                                          03304
             r1 _ save;                                                     03305
             r1.sdaid _ da.dalhandle;                                       03306
             IF NOT SKIP !JSYS strda THEN                                   03307
                IF r1 = syserr THEN NULL                                    03308
                ELSE dismes(2, $"Link Display Error");                      03309
             END;                                                           03310
          END;                                                              03311
    RETURN;                                                                 03312
    END.

                                                                            03313
%...graphics primitives for alphanumeric (line-processor) displays %
                                                                            02796
    (ipdspstr) PROCEDURE (x, y, string, standoutflag);  %display the
```

```
specified string at specified location on a line processor
alpha-numeric display.  IF x = current then use current position of
cursor.  If standoutflag is TRUE then cause the string to standout%
                                                                    02797
   LOCAL store, start, char, l, mx;                                 02798
   LOCAL STRING send[200];                                          02799
   REF string;                                                      02800
   IF x NOT= current THEN position(x, y);                           02801
   IF standoutflag THEN standout();                                 02802
   CCPOS SF(*string*);                                              02803
   start _ store _ chbmty + $send;                                  02804
   *send* _ NULL;                                                   02805
   l _ send.L;  mx _ send.M;                                        02806
   LOOP                                                             02807
      BEGIN                                                         02808
      CASE char _ READC OF                                          02809
         = ENDCHR: EXIT LOOP;                                       02810
         = TAB:                                                     02811
            BEGIN                                                   02812
            ^store _ lptab; BUMP l;  %*send* _ *send*, lptab;%      02813
            END;                                                    02814
         = EOL:                                                     02815
            BEGIN                                                   02816
            !sout(dspjfn, start, -1);                               02817
            *send* _ NULL;                                          02818
            l _ send.L;                                             02819
            store _ start;                                          02820
            IF x NOT= current THEN position(x _ 0, y _ y+1)         02821
            ELSE                                                    02822
               BEGIN                                                02823
               !bout(dspjfn, CR);                                   02824
               !bout(dspjfn, LF);                                   02825
               pad(lpdlpad);                                        02826
               END;                                                 02827
            END;                                                    02828
         = LF:                                                      02829
            BEGIN                                                   02830
            !sout(dspjfn, start, -1);                               02831
            *send* _ NULL;                                          02832
            l _ send.L;                                             02833
            store _ start;                                          02834
            IF x NOT= current THEN position(x+CCPOS, y)             02835
            ELSE                                                    02836
               BEGIN                                                02837
               !bout(dspjfn, LF);                                   02838
               pad(lpdlpad);                                        02839
               END;                                                 02840
            END;                                                    02841
         ENDCASE                                                    02842
            BEGIN                                                   02843
            ^store _ char; BUMP l; %*send* _ *send*, char;%         02844
            END;                                                    02845
      IF l >= mx THEN %time to sout%                                02846
         BEGIN                                                      02847
         !sout(dspjfn, start, -1);                                  02848
         *send* _ NULL;                                             02849
```

```
         l _ send.L;                                                    02850
         store _ start;                                                 02851
         END;                                                           02852
      END;                                                              02853
   IF l THEN %output it%                                                02854
      !sout(dspjfn, start, -l);                                         02855
   IF standoutflag THEN endstandout();                                  02856
   track();                                                             02857
   RETURN;                                                              02858
   END.                                                                 02859

                                                                        02860
(pad) PROCEDURE (count);  %pad with count null characters%              02861
   %adjusts for current line speed by dividing count by lpbaudfactor%
                                                                        02862
   count _ MIN((count/lpbaudfactor) + 1, padstr.M);                     02863
   IF count > 0 THEN                                                    02864
      !sout(dspjfn, chbmty + $padstr, -count);                          02865
   RETURN;                                                              02866
   END.                                                                 02867

                                                                        02868
(position) PROCEDURE % position cursor - alphanumeric displays %
                                                                        02869
   ( xpos,  % x position %                                              02870
     ypos); % y position %                                              02871
   LOCAL STRING send[10];  %for sout%                                   02872
   IF xpos NOT IN [0, lpxmax] OR ypos NOT IN [0, lpymax] THEN           02873
      err($"Illegal coordinate detected in POSITION");                  02874
   *send* _ lpesc, lpposition, xpos+40B, lpymax-ypos+40B;               02875
   tracking _ FALSE;                                                    02876
   !sout(dspjfn, chbmty + $send, -send.L);                              02877
   RETURN;                                                              02878
   END.                                                                 02879

                                                                        02880
(lpttywindow) PROCEDURE % specify tty window - alphanumeric displays
%                                                                       02881
   ( top,    % top line of window %                                     02882
     bottom  % bottom line of window %                                  02883
   );                                                                   02884
   LOCAL save;                                                          02885
   LOCAL STRING send[10];                                               02886
   IF top NOT IN [0, lpymax] OR bottom NOT IN [0, lpymax] THEN          02887
      err($"Illegal coordinate detected in LPTTYWINDOW");               02888
   *send* _ lpesc, lptty, lpymax-top+40B, lpymax-bottom+40B;            02889
   save _ tracking := FALSE; %to avoid printer output%                  02890
   !sout(dspjfn, chbmty + $send, -send.L);                              02891
   tracking _ save; %to avoid printer output%                           02892
   RETURN;                                                              02893
   END.                                                                 02894

                                                                        02895
(lprset) PROCEDURE; % reset alphanumeric displays %                     02896
   LOCAL save;                                                          02897
   LOCAL STRING send[10];                                               02898
```

```
   *send* _ lpesc, lpreset;                                          02899
   save _ tracking := FALSE; %to avoid printer output%               02900
   !sout (dspjfn, chbmty + $send, -send.L);                          02901
   pad(lpdlpad);                                                     02902
   cscreen(); % be sure screen cleared %                             02903
   tracking _ save; %to avoid printer output%                        02904
   RETURN;                                                           02905
   END.                                                              02906

                                                                     02907
(cscreen) PROCEDURE; % clear screen - alphanumeric displays %        02908
   LOCAL save;                                                       02909
   LOCAL STRING send[10];                                            02910
   *send* _ lpesc, lpcscreen;                                        02911
   save _ tracking := FALSE; %to avoid printer output%               02912
   !sout(dspjfn, chbmty + $send, -send.L);                           02913
   pad(lpdlpad);                                                     02914
   tracking _ save; %to avoid printer output%                        02915
   RETURN;                                                           02916
   END.                                                              02917

                                                                     02918
(track) PROCEDURE; % resume tracking - alphanumeric displays %       02919
   LOCAL STRING send[10];                                            02920
   *send* _ lpesc, lptrack;                                          02921
   !sout(dspjfn, chbmty + $send, -send.L);                           02922
   tracking _ TRUE;                                                  02923
   IF tracksend THEN lppsr( tracksend := 0 );                        02924
   RETURN;                                                           02925
   END.                                                              02926

                                                                     02927
(cline) PROCEDURE % write blanks - alphanumeric displays %           02928
   (x,        %x coordinate or CURRENT%                              02929
   y,         %y coordinate or CURRENT%                              02930
   nchar );  % number of blanks to write %                           02931
   LOCAL STRING send[10];                                            02932
   IF nchar NOT IN [1,lpxmax] THEN err($"Illegal number of blanks
   requested in CLINE");                                             02933
   IF x NOT= current THEN position(x, y);                            02934
   *send* _ lpesc, lpcline, nchar+40B;                               02935
   !sout(dspjfn, chbmty + $send, -send.L);                           02936
   pad(nchar);                                                       02937
   IF x NOT= current THEN track();                                   02938
   RETURN;                                                           02939
   END.                                                              02940

                                                                     02941
(dline) PROCEDURE % delete line - alphanumeric displays %            02942
   (x,     %x coordinate or CURRENT%                                 02943
   y);     %y coordinate or CURRENT%                                 02944
   LOCAL STRING send[10];                                            02945
   IF x NOT= current THEN position(x, y);                            02946
   *send* _ lpesc, lpdline;                                          02947
   !sout(dspjfn, chbmty + $send, -send.L);                           02948
   pad(lpdlpad);                                                     02949
```

```
   track();                                                          02950
   RETURN;                                                           02951
   END.                                                              02952

                                                                     02953
(inline) PROCEDURE % insert line - alphanumeric displays %           02954
   (x,             %x coordinate or CURRENT%                         02955
   y,              %y coordinate or CURRENT%                         02956
   string,         %string to be displayed%                          02957
   standoutflag); %FLAG -- TRUE: make the string standout%           02958
   LOCAL STRING send[10];                                            02959
   REF string;                                                       02960
   IF x NOT= current THEN position(x, y);                            02961
   *send* _ lpesc, lpinline;                                         02962
   !sout(dspjfn, chbmty + $send, -send.L);                           02963
   lpdspstr( current, current, &string, standoutflag);               02964
   RETURN;                                                           02965
   END.                                                              02966

                                                                     02967
(lpmarkit) PROCEDURE % bug selection mark - alphanumeric displays %
                                                                     02968
   ( xpos,   % x position of char to bug %                           02969
     ypos); % y position of char to bug %                            02970
   LOCAL save;                                                       02971
   LOCAL STRING send[10];                                            02972
   IF xpos NOT IN [0, lpxmax] OR ypos NOT IN [0, lpymax] THEN        02973
      err($"Illegal coordinate detected in POSITION");               02974
   *send* _ lpesc, lpbug, xpos+40B, lpymax-ypos+40B;                 02975
   save _ tracking := FALSE; %to avoid printer output%               02976
   !sout(dspjfn, chbmty + $send, -send.L);                           02977
   pad(8);                                                           02978
   tracking _ save; %to avoid printer output%                        02979
   RETURN;                                                           02980
   END.                                                              02981

                                                                     02982
(lppopmark) PROCEDURE; % pop bug selection mark - alphanumeric
displays %                                                           02983
   LOCAL save;                                                       02984
   LOCAL STRING send[10];                                            02985
   *send* _ lpesc, lppbug;                                           02986
   save _ tracking := FALSE; %to avoid printer output%               02987
   !sout(dspjfn, chbmty + $send, -send.L);                           02988
   pad(8);                                                           02989
   tracking _ save; %to avoid printer output%                        02990
   RETURN;                                                           02991
   END.                                                              02992

                                                                     02993
(intter) PROCEDURE; % interrogate alphanumeric displays %            02994
   LOCAL save;                                                       02995
   LOCAL STRING send[10];                                            02996
   *send* _ lpesc, lpintter;                                         02997
   save _ tracking := FALSE; %to avoid printer output%               02998
   !sout(dspjfn, chbmty + $send, -send.L);                           02999
```

```
   tracking _ save; %to avoid printer output%                          03000
   RETURN;                                                             03001
   END.                                                                03002

                                                                       03003
(lpcmode) PROCEDURE; % set Lineprocessor to coordinate mode %          03004
   LOCAL save;                                                         03005
   LOCAL STRING send[10];                                              03006
   *send* _ lpesc, lpcoord;                                            03007
   save _ tracking := FALSE; %to avoid printer output%                 03008
   !sout(dspjfn, chbmty + $send, -send.L);                             03009
   tracking _ save; %to avoid printer output%                          03010
   RETURN;                                                             03011
   END.                                                                03012

                                                                       03013
(standout) PROCEDURE; % Send stand out following characters command -
alphanumeric displays %                                                03014
   LOCAL STRING send[10];                                              03015
   *send* _ lpesc, lpstandout;                                         03016
   !sout(dspjfn, chbmty + $send, -send.L);                             03017
   RETURN;                                                             03018
   END.                                                                03019

                                                                       03020
(endstandout) PROCEDURE; % Send end stand out command - alphanumeric
displays %                                                             03021
   LOCAL STRING send[10];                                              03022
   *send* _ lpesc, lpendstandout;                                      03023
   !sout(dspjfn, chbmty + $send, -send.L);                             03024
   RETURN;                                                             03025
   END.                                                                03026

                                                                       03027
(lppopen) PROCEDURE; % open Lineprocessor printer %                    03028
   LOCAL STRING send[10];                                              03029
   *send* _ lpesc, lptptopen;                                          03030
   !sout(dspjfn, chbmty + $send, -send.L);                             03031
   RETURN;                                                             03032
   END.                                                                03033

                                                                       03034
(lppclose) PROCEDURE; % close Lineprocessor printer %                  03035
   LOCAL STRING send[10];                                              03036
   *send* _ lpesc, lptptclose;                                         03037
   !sout(dspjfn, chbmty + $send, -send.L);                             03038
   RETURN;                                                             03039
   END.                                                                03040

                                                                       03041
(lppsr) PROCEDURE % Lineprocessor printer string handler %             03042
   (lppcnt); % number of characters to send %                          03043
   % sends a string of lppcnt chars to Line Processor printer.  Gets
   them from procedure lppch %                                         03044
   LOCAL STRING send[200];                                             03045
   LOCAL store, l;                                                     03046
```

```
   IF &lppch=0 THEN err($"no LP character routine - lppsr");
   IF lppcnt<=0 OR lppjfn=0 THEN RETURN;
   *send* _ NULL;
   l _ send.L;
   store _ chbmty + $send;
   ^store _ lpesc;
   ^store _ lptptsr;
   ^store _ 40B;
   ^store _ lppcnt+40B;
   l _ l + 4;
   WHILE (lppcnt _ lppcnt-1) >= 0 DO
      BEGIN
      ^store _ lppch(); % get character and store it %
      BUMP l;
      END;
   !sout(dspjfn, chbmty+$send, l);
   !gtsts(lppjfn);
   IF r2 .A 1B9 THEN % end of file! %
      lppterm();
   RETURN;
   END.
(lppterm) PROCEDURE; % terminate LP printing %
   % aborts Line Processor printer operation %
   !dti(0); % deassign null %
   !dic(4B5,200B); % deactivate chan 28 %
   lppclose();
   IF NOT SKIP !closf(lppjfn) THEN err($"cant close text file");

   lppjfn _ tracksend _ 0;
   RETURN;
   END.

(lppstart) PROCEDURE % start LP printing from jfn %
   (jfn, % jfn for text file %
   proc); % address of procedure to get chars from %
   % call this procedure to start printing on LP printer %
   % setup variables %
      lppjfn _ jfn;
      &lppch _ proc;
      tracksend _ lppcol _ lpplin _ 0;
   % setup PSI stuff for string requests from LP %
      chntab[28] _ 1B6+$lppint; % use channel 28 for interrupt %

      !ati(0+28); % designate null (zero) as interrupt char %
      !aic(4B5,200B); % activate chn 28 %
   lppopen(); % send open string %
   RETURN;
   END.

(lppint) PROCEDURE; % LP printer interupt routine%
   %---------------------%

   %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%

   %---------------------%
   %save registers%
```

```
svac1 _ r1;                                                    03101
r1 _ $svacs;                                                   03102
!BLT r1, svacse;                                               03103
!MOVE r1,40B; sav40 _ r1;                                      03104
s _ -100; !HRL s,s; !HRRI s,psistk;                            03105
m _ s;                                                         03106
IF tracking THEN lppsr(16)                                     03107
ELSE tracksend _ tracksend + 16;                               03108
%restore registers%                                            03109
r1 _ sav40; !MOVEM r1,40B;                                     03110
!HRLZI r1, svacs;                                              03111
!BLT r1, 17B;                                                  03112
r1 _ svac1;                                                    03113
!debrk;                                                        03114
END.
                                                               03115
```

EXEC FL

```
< NLS, EXECFL.NLS.21, >, 25-Apr-78 13:05 JDH ;;;;
FILE execfl   %  L10  <REL-NLS>EXECFL  %% (110,) (rel-nls,EXECFL.rel,) %
                                                                    01786
%.....Declarations.....%
                                                                    01787
   REGISTER r1=1, r2=2, r3=3, r4=4;                                 01788
%.....file manipulation commands' support routines.....%
                                                                      03
   (parseinput) % parse input file link from user %                   04
      PROCEDURE                                                      05
         (fname,          % adr of user input file name %           02394
         udirname,        % address of string to get directory name %  010
         ufilname,        % address of string to get file name %       011
         uextname,        % address of string to get extension name %  012
         uverno,          % address of string to get version number %  013
         ulftovr,         % address of string to get remaining input % 014
         uname, % address of string to get completed file name %      015
         flags  % bits saying which file fields had *s in them %      016
         );                                                          017
      LOCAL                                                          018
         bytptr % byte pointer for use with jsies %                  019
         ;                                                           020
      LOCAL TEXT POINTER                                             021
         tp1, tp2, tp3, dp1, dp2, fp1, fp2;                          022
      LOCAL STRING                                                   023
         locstr[200]    % local work string %                        024
         ;                                                           025
      REF                                                            026
         fname, udirname, ufilname, uextname, uverno, ulftovr, uname,
         flags;                                                      027
                                                                     028
      % FUNCTION %                                                   029
         % this procedure parses the input file link entered by a user.
                                                                     030
         the following defaults are supplied:                        031
            directory field defaults to null                         032
               (this will force standard TENEX default of connected
               directory to apply.)                                  033
            file name field has no defaults and null is an error     034
            extension name field defaults to NLS if not supplied     035
            version number field defaults to *                       036
               (this will cause all versions of a file to be dealt with)
                                                                     037
            other fields default to null                             038
               (this will force standard TENEX defaults to apply.)   039
         terminating a field with a ^F forces recognition of that
         specific field.                                             040
         terminating a field with a <ALT> forces recognition of that
         specific field and causes remaining unspecified fields to
         default to the above defaults.                              041
         terminating a field with unspecified a space or an <EOL>
         completes that specific field and causes remaining fields to
         default to the above defaults. %                            042
                                                                     043
      % initialize locals %                                          044
         bytptr _ 0;                                                 045
```

```
   % do some initial parsing %                                          02395
    FIND SF(*fname*) ^dp1 ^dp2 ^fp1 SE(*fname*) ^fp2;                   02396
    IF FIND dp1 >                                                       02397
       '< ^dp1 [*> ^dp2 _dp2 / ('<^F>/'<ESC>) ^dp2]  ^fp1 THEN
       NULL;                                                            02398
   % set up local string to work with %                                  046
    *locstr* _ fp1 fp2;                                                 047
   % get directory name from directory input %                           048
    *udirname* _ + dp1 dp2, 0;                                          049
    BUMP DOWN udirname.L;  % discount null byte at end %                050
    IF *udirname*[udirname.L] = $ascalt                                 051
       THEN *udirname*[udirname.L] _ $ctlf;                             052
    % default to connected directory if not specified %                 053
       IF NOT udirname.L THEN                                           054
          BEGIN    % get connected directory name string %              055
          !gjinf();                                                     056
          gdname( r2, &udirname);                                       057
          *udirname* _ *udirname*, 0;                                   058
          BUMP DOWN udirname.L;    % discount null byte at end %
                                                                        059
          END;                                                          060
   % get file name from input file name %                                061
    FIND > SF(*locstr*) ^tp1 [ ('. / '; / '<^F> / '<ESC> / SP /
    EOL) ^tp2 _tp2 / ENDCHR ^tp2] ^tp3 ;                                062
    CCPOS tp2;                                                          063
    CASE READC OF                                                       064
       = '.:                                                            065
          BEGIN                                                         066
          *ufilname* _ tp1 tp2, 0;                                      067
          BUMP DOWN ufilname.L;                                         068
          END;                                                          069
       = ';:                                                            070
          BEGIN                                                         071
          *ufilname* _ tp1 tp2, 0;                                      072
          BUMP DOWN ufilname.L;                                         073
          *uextname* _ "NLS", 0;                                        074
          BUMP DOWN uextname.L;                                         075
          GOTO gtver;                                                   076
          END;                                                          077
       = '<^F>:                                                         078
          BEGIN                                                         079
          *ufilname* _ tp1 tp3, 0;                                      080
          BUMP DOWN ufilname.L;                                         081
          % file name of ^F only is equivalent to file name of * %
                                                                        082
             IF ufilname.L = 1 THEN *ufilname*[1] _ '*;                 083
          END;                                                          084
       = '<ESC>:                                                        085
          BEGIN                                                         086
          *ufilname* _ tp1 tp2, '<^F>, 0;                               087
          BUMP DOWN ufilname.L;                                         088
          % file name of only <ALT> is equivalent to *.NLS;* %          089
             IF ufilname.L = 1 THEN *ufilname*[1] _ '*;                 090
          *uextname* _ "NLS", 0;                                        091
          BUMP DOWN uextname.L;                                         092
          *uverno* _ '*, 0;                                             093
```

```
            BUMP DOWN uverno.L;                                          094
            GOTO gtlftovr;                                               095
            END;                                                         096
         = SP,                                                           097
         = EOL:                                                          098
            BEGIN                                                        099
            *ufilname* _ tp1 tp2, 0;                                    0100
            BUMP DOWN ufilname.L;                                       0101
            *uextname* _ "NLS", 0;                                      0102
            BUMP DOWN uextname.L;                                       0103
            *uverno* _ '*, 0;                                           0104
            BUMP DOWN uverno.L;                                         0105
            GOTO gtflgs;                                                0106
            END;                                                        0107
         = ENDCHR:                                                      0108
            BEGIN                                                       0109
            *ufilname* _ tp1 tp3, 0;                                    0110
            BUMP DOWN ufilname.L;                                       0111
            *uextname* _ "NLS", 0;                                      0112
            BUMP DOWN uextname.L;                                       0113
            *uverno* _ '*, 0;                                           0114
            BUMP DOWN uverno.L;                                         0115
            GOTO gtflgs;                                                0116
            END;                                                        0117
         ENDCASE err( $"system screwup" );                              0118
      % get extension name from input file name %                       0119
         (parsl1):                                                     03180
         FIND > tp3 ^tp1 [ ('; / '. / '<^F> / '<ESC> / SP / EOL) ^tp2
         _tp2 / ENDCHR ^tp2] ^tp3 ;                                     0120
         CCPOS tp2;                                                     0121
         CASE READC OF                                                  0122
            = ';, = fvrdchar:                                           0123
            BEGIN                                                       0124
            *uextname* _ tp1 tp2, 0;                                    0125
            BUMP DOWN uextname.L;                                       0126
            END;                                                        0127
            = '.: GOTO parsl1; %got second period but not running
            tops20%                                                    03181
            = '<^F>:                                                    0128
            BEGIN                                                       0129
            *uextname* _ tp1 tp3, 0;                                    0130
            BUMP DOWN uextname.L;                                       0131
            % ext name of ^F is equivalent to ext name of NLS %         0132
               IF uextname.L = 1 THEN                                   0133
                  BEGIN                                                 0134
                  *uextname* _ "NLS", 0;                                0135
                  BUMP DOWN uextname.L;                                 0136
                  END;                                                  0137
            END;                                                        0138
            = '<ESC>:                                                   0139
            BEGIN                                                       0140
            *uextname* _ tp1 tp2, '<^F>, 0;                             0141
            BUMP DOWN uextname.L;                                       0142
            % ext name of only <ALT> is equivalent to *.NLS;* %         0143
               IF uextname.L = 1 THEN                                   0144
                  BEGIN                                                 0145
```

```
                *uextname* _ "NLS", 0;                                  0146
                BUMP DOWN uextname.L;                                   0147
                END;                                                    0148
            *uverno* _ '*, 0;                                           0149
            BUMP DOWN uverno.L;                                         0150
            GOTO gtlftovr;                                              0151
            END;                                                        0152
         = SP,                                                          0153
         = EOL:                                                         0154
            BEGIN                                                       0155
            *uextname* _ tp1 tp2, 0;                                    0156
            BUMP DOWN uextname.L;                                       0157
            *uverno* _ '*, 0;                                           0158
            BUMP DOWN uverno.L;                                         0159
            GOTO gtflgs;                                                0160
            END;                                                        0161
         = ENDCHR:                                                      0162
            BEGIN                                                       0163
            *uextname* _ tp1 tp3, 0;                                    0164
            BUMP DOWN uextname.L;                                       0165
            *uverno* _ '*, 0;                                           0166
            BUMP DOWN uverno.L;                                         0167
            GOTO gtflgs;                                                0168
            END;                                                        0169
         ENDCASE err( $"system screwup" );                              0170
   % get version number string from input file name %                   0171
      (gtver):                                                          0172
      IF FIND > tp3 ^tp1 $1'- 1$D ^tp2 ^tp3 THEN                        0173
         BEGIN                                                          0174
         *uverno* _ tp1 tp2, 0;                                         0175
         BUMP DOWN uverno.L;                                            0176
         END                                                            0177
      ELSE                                                              0178
         BEGIN                                                          0179
         *uverno* _ '*, 0;                                              0180
         BUMP DOWN uverno.L;                                            0181
         IF FIND > tp1 ^tp3 ('<^F> / '<ESC> / '* ^tp3) THEN NULL;
                                                                        0182
         END;                                                           0183
   % now get any remaining user input %                                 0184
      (gtlftovr):                                                       0185
      FIND > tp3 ^tp1 $1CH ^tp2 [ENDCHR] ^tp3 ;                         0186
      CCPOS tp1;                                                        0187
      CASE READC OF                                                     0188
         = ';, ='<^F>, ='<ESC>:                                         0189
            BEGIN                                                       0190
            *ulftovr* _ tp2 tp3, 0;                                     0191
            BUMP DOWN ulftovr.L;                                        0192
            END;                                                        0193
         ENDCASE                                                        0194
            BEGIN                                                       0195
            *ulftovr* _ tp1 tp3, 0;                                     0196
            BUMP DOWN ulftovr.L;                                        0197
            END;                                                        0198
   % now set flags for which fields (if any) the user entered a * %
                                                                        0199
```

```
    (gtflgs):                                                          0200
    flags.dstar _                                                      0201
       IF udirname.L = 1 AND *udirname*[1] = '* THEN TRUE ELSE
       FALSE;                                                          0202
    flags.fstar _                                                      0203
       IF ufilname.L = 1 AND *ufilname*[1] = '* THEN TRUE ELSE
       FALSE;                                                          0204
    flags.estar _                                                      0205
       IF uextname.L = 1 AND *uextname*[1] = '* THEN TRUE ELSE
       FALSE;                                                          0206
    flags.vstar _                                                      0207
       IF uverno.L = 1 AND *uverno*[1] = '* THEN TRUE ELSE FALSE;
                                                                       0208
% now build completed name %                                          0209
    IF (FIND SF(*udirname*)  ['<] ['>] ) THEN *uname* _ *udirname*
    ELSE                                                             03224
       BEGIN                                                         03225
       *uname* _ '<, *udirname*;                                       0210
       IF *uname*[uname.L] # '<^F> THEN *uname* _ *uname*, '>;          0211
       END;                                                          03226
    *uname* _ *uname*, *ufilname*;                                     0212
    IF *uname*[uname.L] # '<^F> THEN *uname* _ *uname*, '.;             0213
    *uname* _ *uname*, *uextname*;                                     0214
    IF *uname*[uname.L] # '<^F> THEN *uname* _ *uname*, fvrdchar;
                                                                       0215
    *uname* _ *uname*, *uverno*;                                       0216
    IF ulftovr.L # 0 THEN *uname* _ *uname*, ';, *ulftovr*;            0217
    *uname* _ *uname*, 0;                                              0218
    BUMP DOWN uname.L;                                                 0219
% all done now (i hope) %                                             0220
    RETURN;                                                            0221
                                                                       0222
END.
                                                                       0223
%  %                                                                 0224
```

```
(unprseinpt) % parse jfn into individual strings %                    0225
   PROCEDURE                                                          0226
      (fjfn,          % jfn of concerned file %                       0227
      jfnflgs,              % left-half bits returned by gtjfn %
                                                                      0228
      udirname,       % address of string to receive directory name %
                                                                      0229
      ufilname,       % address of string to receive file name %     0230
      uextname,       % address of string to receive extension name %
                                                                      0231
      uverno,               % address of string to receive version
      number %                                                        0232
      ulftovr,              % address of string to receive remaining
      stuff %                                                         0233
      uname,          % address of string to receive complete name %
                                                                      0234
      flags           % address of cell to receive * indicating bits %
                                                                      0235
      );                                                              0236
   LOCAL                                                              0237
      fuljfn          % jfn and flags %                               0238
      ;                                                               0239
   LOCAL TEXT POINTER                                                 0240
      tp1, tp2, tp3, tp4, tp5, tp6, tp7, tp8, tp9, tp10;              0241
   REF udirname, ufilname, uextname, uverno, ulftovr, uname, flags;
                                                                      0242
                                                                      0243
   % initialize locals %                                             0244
      fuljfn _ 0;                                                     0245
   % get complete name from jfns %                                   0246
      fuljfn.LH _ jfnflgs;                                            0247
      fuljfn.RH _ fjfn;                                               0248
      jfntostr( fuljfn, &uname, 011111140001B);                       0249
   % set up pairs of text pointers to each field %                   0250
      IF NOT FIND SF(*uname*) > ['<] < CH (': $LD / ) > ^tp1 ['>]
      ^tp2 ^tp3 ['.] ^tp4 _tp4 ^tp5 ['; / '.] ^tp6 _tp6 ^tp7 ['; ]
      ^tp8 _tp8 ^tp9 [ENDCHR] ^tp10 THEN err( $"system screwup");
                                                                      0251
   % now build the individual strings %                              0252
      *udirname* _ + tp1 tp2, 0;  BUMP DOWN udirname.L;               0253
      *ufilname* _ + tp3 tp4, 0;  BUMP DOWN ufilname.L;               0254
      *uextname* _ + tp5 tp6, 0;  BUMP DOWN uextname.L;               0255
      *uverno* _ + tp7 tp8, 0;  BUMP DOWN uverno.L;                   0256
      *ulftovr* _ + tp9 tp10, 0;  BUMP DOWN ulftovr.L;                0257
   % now setup flags indicating which fields were astericks %        0258
      flags.dstar _ jfnflgs.dstar;                                    0259
      flags.fstar _ jfnflgs.fstar;                                    0260
      flags.estar _ jfnflgs.estar;                                    0261
      flags.vstar _ jfnflgs.vstar;                                    0262
   % all done so return %                                            0263
      RETURN;                                                         0264
   END.
                                                                      0265
   %  %                                                              0266
```

```
(gthstn)     % returns host number for input string text pointers %
                                                                    0267
   PROCEDURE                                                        0268
      (hp1,  % text pointer to start of host name %                 0269
      hp2    % text pointer to end of host name %                   0270
      );                                                            0271
   LOCAL                                                            0272
      ihostn,         % index for looping through HOSTN table %     0273
      sindex,         % index into HSTNAM table ( gotten from HOSTN
      table) %                                                      0274
      jhstnm,         % index for getting asciz string from HSTNAM
      table %                                                       0275
      rhstn, % remote host number for an entry in HOSTN table %     0276
      kindex % index for doing string word compares %               0277
      ;                                                             0278
   LOCAL STRING                                                     0279
      rhoststr[45],  % input host name string %                     0280
      tstring[45]    % temp string for HSTNAM table asciz string %
                                                                    0281
      ;                                                             0282
   REF hp1, hp2;                                                    0283
                                                                    0284
   % ALGORITHM %                                                   0285
      % if the input text pointers point to a null string then the
      host number of the local host is reaturned.                   0286
      This procedure also sets up the following globals:            0287
         hostni.LH - gets table number of HOSTN table for getab jsys
                                                                    0288
         hostni.RH - gets number of entries in HOSTN table          0289
         hostsi.LH - gets table number of HSTNAM table for getab jsys
                                                                    0290
         hostsi.RH - gets number of entries in HSTNAM table         0291
      %                                                             0292
   % RETURNS %                                                     0293
      % returns host number for input texp pointers (local host
      number if text pointers delimit a null string) or -1 for
      invalid host name input %                                     0294
                                                                    0295
   % setup string for input text pointers %                        0296
      *rhoststr* _ + hp1 hp2;                                       0297
   % if null input then return local host number %                 0298
      IF NOT rhoststr.L THEN RETURN( lhostn );                      0299
   % setup string for word compares later %                        0300
      *rhoststr* _ *rhoststr*, 0, 0, 0, 0, 0, 0;                    0301
      rhoststr.L _ rhoststr.L - 6;                                  0302
   % setup hostni with HOSTN number and length %                   0303
      IF hostni < 0 THEN                                            0304
         BEGIN                                                      0305
         !sysgt( r1 _ getsbn( $"HOSTN" ) );                         0306
         IF NOT r2 THEN err( $"sysgt jsys error" );                 0307
         hostni.LH _ r2.RH;  % HOSTN table number %                 0308
         !HLRE r2,r2;                                               0309
         hostni.RH _ -r2;    % number of entries %                   0310
         END;                                                       0311
   % setup hostsi with HSTNAM number and length %                  0312
      IF hostsi < 0 THEN                                            0313
```

```
        BEGIN                                                          0314
        !sysgt( r1 _ getsbn( $"HSTNAM" ) );                            0315
        IF NOT r2 THEN err( $"sysgt jsys error" );                     0316
        hostsi.LH _ r2.RH;  % HSTNAM table number %                    0317
        !HLRE r2,r2;                                                   0318
        hostsi.RH _ -r2;    % number of entries %                       0319
        END;                                                           0320
    % loop through HOSTN table looking for this host string %          0321
      FOR ihostn _ hostni.RH DOWN UNTIL <= 0 DO                        0322
        BEGIN                                                          0323
        r1.LH _ ihostn - 1; % table starts at 0 %                      0324
        r1.RH _ hostni.LH;                                             0325
        IF NOT SKIP !getab( r1 ) THEN err( $"getab jsys error" );
                                                                       0326
        % get index into HSTNAM and host # this entry %                0327
          sindex _ r1.hstnmi;                                          0328
          rhstn _ r1.hstnmn;                                           0329
        % now get asciz host name string from HSTNAM %                 0330
          FOR jhstnm _ sindex UP UNTIL < 0 DO                          0331
            BEGIN                                                      0332
            r1.LH _ jhstnm;                                            0333
            r1.RH _ hostsi.LH;                                         0334
            IF NOT SKIP !getab( r1 ) THEN err( $"getab jsys error"
            );                                                         0335
            tstring[jhstnm - sindex + 1] _ r1;                         0336
            IF NOT ( r1 .A 377B ) THEN EXIT LOOP;                      0337
            END;                                                       0338
        % see if this is the desired host %                            0339
          FOR kindex _ 1 UP UNTIL > (jhstnm - sindex + 1) DO           0340
            IF (rhoststr[kindex] .A 777777777776B) #
            tstring[kindex] THEN REPEAT LOOP 2;   % look at next
            entry %                                                    0341
        % if we're here we got the right host so return %              0342
          RETURN( rhstn );                                             0343
        END;                                                           0344
    % got here means we've exhausted HOSTN and not found a match %
                                                                       0345
      RETURN( -1 );                                                    0346
                                                                       0347
    END.
                                                                       0348
    %  %                                                              0349
```

```
(chkdev)     % returns FALSE if not a supported (disk) file %     0350
   PROCEDURE                                                      0351
      (fjfn  % jfn of file to check %                             0352
      );                                                          0353
   LOCAL STRING                                                   0354
      devstring[40]  % string to get device string for this jfn %
                                                                  0355
      ;                                                           0356
                                                                  0357
   % get device name string from jfns %                           0358
      jfntostr(fjfn, $devstring, 100000B6);                       0359
   % now see if a disk file %                                     0360
      RETURN( IF (*devstring* = "DSK") OR (*devstring* = "PS") THEN
      TRUE ELSE FALSE );                                          0361
                                                                  0362
   END.
                                                                  0363
   %  %                                                          0364
```

```
(chkpcs)     % check if * for extname and extname = "PC" %          0365
   PROCEDURE                                                       0366
      (ijfn, % jfn of file to check %                              0367
      flags  % left half jfn flags of input %                      0368
      );                                                           0369
   LOCAL STRING                                                    0370
      pcname[40]     % temp string to get actual extension name %
                                                                   0371
      ;                                                            0372
                                                                   0373
   % RETURNS %                                                     0374
      % returns FALSE if * was input for the extension name and the
      extension name is "PC" (i.e. this file is a partial copy);
      returns true otherwise %                                     0375
                                                                   0376
   % find out if * was input for extension name %                  0377
      IF NOT flags.estar THEN RETURN( TRUE );                      0378
   % now get the actual extension name %                          0379
      jfntostr( fjfn, $pcname, 000100B6 );                         0380
   % now check for extension name of "PC" %                       0381
      IF *pcname* = "PC" THEN RETURN( FALSE ) ELSE RETURN( TRUE );
                                                                   0382
                                                                   0383
   END.
                                                                   0384
   %  %                                                           0385
```

```
(getpcjfn)   % get jfn for pc of passed jfn %                          0410
   PROCEDURE                                                           0411
      (fjfn, % jfn of file %                                           0412
      pcjfn, % address of variable to receive PC jfn %                 0413
      errstring,     % address of string to receive error messages %
                                                                       0414
      isok           % TRUE: ok to get pc if locked by someone else %
                                                                       0415
      );                                                               0416
   LOCAL                                                               0417
      temp,          % temp for pc name creation %                    02317
      dircode,       % lock code from file uws %                      03227
      dirnum,        % directory number of locking user %              0418
      init   % ident of locking user %                                 0419
      ;                                                                0421
   LOCAL STRING                                                        0422
      dirstring[40], % temp string to check for valid locking dir. #
      %                                                                0423
      fstring[200],  % string to get file name for passed file jfn %
                                                                       0424
      pcstring[200]  % string to get partial copy name %               0425
      ;                                                                0426
   REF pcjfn, errstring;                                               0427
                                                                       0428
   % FUNCTION %                                                       0429
      % get a jfn for the partial copy (if one exists) for the passed
      jfn %                                                            0430
   % RETURNS %                                                        0431
      % returns TRUE if no partial copy exists, or if partial copy
      exists and the file is locked by this user, or if partial copy
      exists and isok parameter is TRUE, or if a partial copy exists
      and this user is an enabled wheel, or if the file is the TOPS20
      MAIL.TXT file (which uses the user-settable word); returns
      FALSE otherwise. %                                               0432
                                                                       0433
   % initialize locals %                                              0434
      dirnum _ init _ 0;                                               0435
   % initialize error string %                                       02399
      *errstring* _ NULL;                                              02400
   % initially there is no pc jfn %                                   0436
      pcjfn _ 0;                                                       0437
   % now get user settable word to determine if pc exists %           0438
      !gtfdb( fjfn, 1B6+24B, $r3);                                     0439
   % return true if no pc %                                           0440
      IF (dircode _ r3.lkdirn) = 0 THEN RETURN( TRUE );                0441
   % get ident of locking user %                                       0442
      init _ r3.lkinit;                                                0443
   % orig. filename %                                                 03230
      jfnstr(fjfn, $fstring);                                          03185
   % check if MAIL.TXT file on TOPS20 %                               03182
      IF tops20flag THEN                                               03183
         BEGIN                                                         03184
         IF FIND SF(*fstring*) ["">MAIL.TXT."] THEN RETURN(TRUE);
                                                                       03186
         END;                                                          03187
   % make pc file name %                                             03228
```

```
   apctilnam($pcstring, $fstring);                                    03229
% pc directory %                                                      03231
   IF NOT ( dirnum _ drcdecode(dircode, $pcstring, init)) THEN
   RETURN(TRUE);                                                      03232
% see if locked by this user %                                       0447
   IF NOT ( (dirnum = logdirno) AND (init = cinit) ) THEN             0448
      BEGIN       % locked by someone else %                        0449
      % if invalid locking dir. # then not an NLS file, return
      TRUE %                                                         0450
         IF NOT SKIP !dirst( chbmty+$dirstring, dirnum) THEN
         RETURN( TRUE );                                             0451
      % ok to let user have this file even tho locked ? %           0452
         IF NOT isok THEN                                            0453
            % find out if this is an enabled wheel %                0454
               IF NOT isenabled() THEN                               0455
                  BEGIN   % not an enabled wheel %                   0456
                  % create error message %                          0457
                     lockid ( &errstring, dirnum, init);            0458
                  % now return FALSE %                              0459
                     RETURN( FALSE );                               0460
                  END;                                              0461
      END;                                                          0462
% now get string name for partial copy %                           0463
   temp _ cpcnam( $fstring, dirnum);                                0465
% now get a jfn for the partial copy %                             0466
   pcjfn _ sgtjfn( gtjprv+gtjoif+gtjidl, temp, &errstring);         02318
% free the string obtained in getstring and return %               02319
   freestring( temp, $dspblk);                                      02320
   RETURN( pcjfn );                                                 0467
END.

                                                                    0468
%  %                                                                0469
```

```
(isenabled)  % determine if this user is an enabled wheel %          0470
   PROCEDURE;                                                        0471
                                                                     0472
   % RETURNS %                                                      0473
      % returns TRUE if this user is an enabled wheel; FALSE
      otherwise %                                                    0474
                                                                     0475
   % get enabled capabilites for this process %                     0476
      !rpcap( 400000B );                                             0477
   % now return true or false depending on state of this user %     0478
      IF r3 .A 4B5 THEN RETURN( TRUE ) ELSE RETURN( FALSE );         0479
                                                                     0480
   END.
                                                                     0481
   %  %                                                             0482
```

```
(delfiandpc) % delete file (and partial copy) for input jfns %
   PROCEDURE
      (fjfn, % jfn of file %
      pcjfn, % jfn of PC (or 0) %
      erstrng        % address of string to receive error messages %
      );
   LOCAL
      arstus,        % TRUE means archive pending for the file %
      delmode        % FALSE means pc was already deleted %
      ;
   REF erstrng;

   % ALGORITHM %
      % deletes files by changing the deleted bit in the FDB %
   % RETURNS %
      % returns TRUE if file(s) deleted properly.  On FALSE returns,
      writes erstrng with reason for failure and nothing is deleted %

   % initialize locals %
      delmode _ 0;
   % delete partial copy first if it exists %
      IF pcjfn THEN
         BEGIN       % find out if already deleted %
         !gtfdb( pcjfn, 1B6 + 1B, $r3);
         IF NOT r3 .A 4B10 THEN
            BEGIN    % not deleted yet %
            delmode _ TRUE;
            !gtfdb( pcjfn, 1B6+$fdbbkf, $r3);
            IF r3.flarfl.fdbarc THEN
               BEGIN % can't delete archive pending partial copy %
               *erstrng* _ "can't delete archive pending partial
               copy";
               RETURN( FALSE );
               END;
            IF NOT chnfdb( pcjfn, 1B, 4B10, 4B10) THEN
               BEGIN % can't delete partial copy %
               *erstrng* _ "can't delete partial copy";
               RETURN( FALSE );
               END;
            END
         ELSE delmode _ FALSE;       % already deleted %
         END;
   % now delete the file itself %
      !gtfdb( fjfn, 1B6+$fdbbkf, $r3);
      arstus _ r3.flarfl.fdbarc;
      IF arstus OR NOT chnfdb( fjfn, 1B, 4B10, 4B10) THEN
         BEGIN       % can't delete the file %
         IF arstus THEN
            *erstrng* _ "can't delete archive pending file"
         ELSE *erstrng* _ "can't delete this file";
         % now restore partial copy to previous state %
            IF delmode THEN
```

```
            % we deleted the partial copy above %                    0522
              IF NOT chnfdb( pcjfn, 1B, 4B10, 0) THEN                0523
                 % we could delete above, but cant undelete here
                 %                                                   0524
                 err( $"system screwup" );                           0525
       % now give false return since we cant delete the file %       0526
          RETURN( FALSE );                                           0527
       END;                                                          0528
% we're all done now so return true %                               0529
   RETURN( TRUE );                                                  0530
                                                                    0531
END.

                                                                    0532
%  %                                                               0533
```

```
(undflandpc) % undelete file (and partial copy) for input jfns %
   PROCEDURE
      (fjfn, % jfn of file %
      pcjfn, % jfn of PC (or 0) %
      erstrng        % address of string to receive error messages %
      );
   LOCAL
      delmode        % FALSE means pc was already undeleted %
      ;
   REF errstring;

   % ALGORITHM %
      % undeletes files by changing the deleted bit in the FDB %
   % RETURNS %
      % returns TRUE if file(s) undeleted properly.  On FALSE
      returns, writes erstrng with reason for failure and nothing is
      undeleted %

   % initialize locals %
      delmode _ 0;
   % undelete partial copy first if it exists %
      IF pcjfn THEN
         BEGIN       % find out if already undeleted %
         !gtfdb( pcjfn, 1B6 + 1B, $r3);
         IF r3 .A 4B10 THEN
            BEGIN    % not undeleted yet %
            delmode _ TRUE;
            IF NOT chnfdb( pcjfn, 1B, 4B10, 0) THEN
               BEGIN % can't undelete partial copy %
               *erstrng* _ "can't undelete partial copy";
               RETURN( FALSE );
               END;
            END
         ELSE delmode _ FALSE;       % already undeleted %
         END;
   % now undelete the file itself %
      IF NOT chnfdb( fjfn, 1B, 4B10, 0) THEN
         BEGIN       % can't undelete the file %
         *erstrng* _ "can't undelete this file";
         % now restore partial copy to previous state %
            IF delmode THEN
               % we undeleted the partial copy above %
                  IF NOT chnfdb( pcjfn, 1B, 4B10, 4B10) THEN
                     % we could undelete above, but cant delete here
                     %
                     err( $"system screwup" );
         % now give false return since we cant undelete the file %

            RETURN( FALSE );
         END;
   % we're all done now so return true %
      RETURN( TRUE );

   END.
```

% %

```
(triflandpc) % trim file (and partial copy) for input jfns %      0585
   PROCEDURE                                                      0586
      (fjfn, % jfn of file %                                      0587
      pcjfn, % jfn of PC (or 0) %                                 0588
      nver,  % number of versions to keep for each file %         0589
      erstrng        % address of string to receive error messages %
                                                                  0590
      );                                                          0591
   LOCAL                                                          0592
      lcount,         % loop index for untrimming %              0593
      pcgjfn,         % group jfn for untrimming partial copies % 0594
      jfnflgs,        % left half gtjfn flags %                   0595
      delmode         % FALSE means pc was already deleted %      0596
      ;                                                           0597
   LOCAL STRING                                                   0598
      jfnname[200],   % string for partial copy untrimming %     0599
      xstring[100]    % error string for sgtjfn %                 0600
      ;                                                           0601
   REF erstrng;                                                   0602
                                                                  0603
   % ALGORITHM %                                                 0604
      % TRIMS files by using the DELNF jsys %                    0605
   % RETURNS %                                                   0606
      % returns TRUE (and number of files deleted) if file(s) trimmed
      properly.  On FALSE returns, writes erstrng with reason for
      failure and nothing is trimmed %                            0607
                                                                  0608
   % initialize locals %                                         0609
      delmode _ 0;                                              0610
   % initialize error string %                                  02401
      *erstrng* _ NULL;                                         02402
   % trim partial copy first if it exists %                     0611
      IF pcjfn THEN                                             0612
         IF NOT SKIP !delnf( pcjfn, nver) THEN                  0613
            BEGIN                                               0614
            *erstrng* _ "can't trim partial copy";              0615
            RETURN( FALSE, 0);                                  0616
            END                                                 0617
         ELSE delmode _ (IF tops20flag THEN r2 ELSE -r2); %number of
         versions deleted %                                     0618
   % now trim the file itself and return %                      0619
      IF NOT SKIP !delnf( fjfn, nver) THEN                      0620
         BEGIN       % can't trim the file %                    0621
         *erstrng* _ "can't trim this file";                    0622
         % now restore partial copy to previous state %         0623
            IF delmode > 0 THEN                                 0624
               BEGIN % we trimmed the partial copy above %      0625
               % get string name for undelete loop %            0626
                  jfntostr( pcjfn, $jfnname, 011100B6+1);       0627
                  *jfnname* _ *jfnname*, fvrdchar, "*", 0;      0628
                  BUMP DOWN jfnname.L;                          0629
               % now get a group jfn for it %                   0630
                  IF NOT (pcgjfn _ sgtjfn( gtjoif .V gtjstr,
                  $jfnname, $xstring : jfnflgs) ) THEN          0631
                     BEGIN                                      0632
                     *erstrng* _ *erstrng*, CR, LF, "   ***   can't
```

```
                    untrim trimmed partial copies ***";          0633
                    RETURN( FALSE, 0 );                          0634
                    END;                                         0635
                % now loop past untrimmed files first %          0636
                  FOR lcount _ 1 UP UNTIL >= nver DO             0637
                    BEGIN                                        0638
                    r1.LH _ jfnflgs;                             0639
                    r1.RH _ pcgjfn;                              0640
                    IF NOT SKIP !gnjfn( r1 ) THEN                0641
                      BEGIN                                      0642
                      IF NOT SKIP !rljfn( pcgjfn ) THEN NULL;    0643
                      *erstrng* _ *erstrng*, CR, LF, "   ***
                      can't untrim trimmed partial copies ***";
                                                                 0644
                      RETURN( FALSE, 0);                         0645
                      END;                                       0646
                    END;                                         0647
                % now untrim partial copies trimmed above %      0648
                  FOR lcount _ 1 UP UNTIL > delmode DO           0649
                    BEGIN                                        0650
                    r1.LH _ jfnflgs;                             0651
                    r1.RH _ pcgjfn;                              0652
                    IF NOT SKIP !gnjfn( r1 ) THEN                0653
                      BEGIN                                      0654
                      IF NOT SKIP !rljfn( pcgjfn ) THEN NULL;    0655
                      RETURN( FALSE, 0 );                        0656
                      END;                                       0657
                    % now untrim one version %                   0658
                      IF NOT chnfdb( pcgjfn, 1B, 4B10, 0) THEN   0659
                        *erstrng* _ *erstrng*, CR, LF, "   ***
                        can't untrim trimmed partial copies ***";
                                                                 0660
                    END;                                         0661
                END;                                             0662
            % now give false return since we cant undelete the file %
                                                                 0663
              RETURN( FALSE, 0 );                                0664
            END                                                  0665
          ELSE RETURN( TRUE, (IF tops20flag THEN r2 ELSE -r2));  0666
                                                                 0667
END.
                                                                 0668
%  %                                                            0669
```

```
(profiandpc) % set protection of a file (and its pc) %                  02321
   PROCEDURE                                                            02322
      (fjfn, % jfn of the file %                                        02323
      pcjfn, % jfn of the pc or zero %                                  02324
      mask,  % mask indicating which bits to change %                   02325
      prot,  % new protection bits %                                    02326
      astr   % address of string to get error message %                 02327
      );                                                                02328
   LOCAL                                                                02329
      newprot,       %computed new protection%                          03068
      pcprot % old protection of the pc %                               02330
      ;                                                                 02331
   REF astr;                                                            02332
                                                                        02333
   % initial variables %                                               02334
      pcprot _ 0;                                                       02335
   % mask only the right half of the protection word %                  03069
      mask.LH _ 0;                                                      03070
   % change protection of the partial copy first %                      02336
      IF pcjfn THEN                                                     02337
         BEGIN                                                          02338
         % get the old protection %                                     02339
            !gtfdb( pcjfn, 1000004B, $pcprot);                          02340
         % change the protection %                                     02341
            IF pcprot.LH # 500000B THEN                                 02342
               BEGIN % if fancy protection reset first %                02343
               newprot  _ (500000777752B .A (mask .X -1)) .V (prot .A
               mask);                                                   02344
               IF NOT chnfdb( pcjfn, 4, 36M, newprot) THEN              02345
                  BEGIN                                                 02346
                  *astr* _ "can't change protection of partial copy";
                                                                        02347
                  RETURN( FALSE );                                      02348
                  END;                                                  02349
               END                                                      02350
            ELSE                                                        02351
               IF NOT chnfdb( pcjfn, 4, mask, prot) THEN                02352
                  BEGIN                                                 02353
                  *astr* _ "can't change protection of partial copy";
                                                                        02354
                  RETURN( FALSE );                                      02355
                  END;                                                  02356
         END;                                                           02357
   % now do the file %                                                 02358
      % get the old protection %                                       02359
         !gtfdb( fjfn, 1000004B, $newprot);                             02360
      % change the protection %                                       02361
         IF newprot.LH # 500000B THEN                                   02362
            BEGIN    % if fancy protection reset first %               02363
            newprot  _ (500000777752B .A (mask .X -1)) .V (prot .A
            mask);                                                      02364
            IF NOT chnfdb( fjfn, 4, 36M, newprot) THEN                  02365
               BEGIN                                                    02366
               *astr* _ "can't change protection of the file";          02367
               % reset the pc protection if changed above %             02368
                  IF pcprot THEN                                        02369
```

```
                    BEGIN                                                 02370
                    IF NOT chnfdb( pcjfn, 4, 36M, pcprot) THEN            02371
                       *astr* _ "file protection not changed;
                       partial copy protection changed";                  02372
                    END;                                                  02373
                RETURN( FALSE );                                          02374
                END;                                                      02375
             END                                                          02376
          ELSE                                                            02377
             IF NOT chnfdb( fjfn, 4, mask, prot) THEN                     02378
             BEGIN                                                        02379
             *astr* _ "can't change protection of the file";              02380
             % reset the pc protection if changed above %                 02381
                IF pcprot THEN                                            02382
                BEGIN                                                     02383
                IF NOT chnfdb( pcjfn, 4, 36M, pcprot) THEN                02384
                   *astr* _ "file protection not changed;
                   partial copy protection changed";                      02385
                END;                                                      02386
             RETURN( FALSE );                                             02387
             END;                                                         02388
% all done, so return %                                                  02389
   RETURN( TRUE );                                                        02390
                                                                          02391
END.
                                                                          02392
%  %                                                                     02393
```

```
(arcflandpc) % set archive status of a file (and its pc) %
   PROCEDURE
      (fjfn, % jfn of the file %
      pcjfn, % jfn of the pc or zero %
      arcparms,      % new archive status bits %
      astr   % address of string to get error message %
      );
   LOCAL
      mask,  % mask indicating which bits to change %
      arstus,        % old archive status of the file %
      pcarc  % old archive status of the pc %
      ;
   REF astr;

   % initial variables %
      *astr* _ NULL;
      pcarc _ 0;
   % set up mask for which bits we will change %
      mask _ FALSE;
      mask.flarfl.fdbadl _ TRUE;
   % change archive status of the partial copy first %
      IF pcjfn THEN
         BEGIN
         % get the old archive status %
            !gtfdb( pcjfn, 186+$fdbbkf, $pcarc);
         % change the archive status %
            IF pcarc.flarfl.fdbaar THEN
               BEGIN
               mask.flarfl.fdbnar _ FALSE;
               mask.flarfl.fdbarc _ FALSE;
               *astr* _ "partial copy already archived";
               END
            ELSE
               BEGIN
               mask.flarfl.fdbnar _ TRUE;
               mask.flarfl.fdbarc _ TRUE;
               END;
            IF NOT chnfdb( pcjfn, $fdbbkf, mask, arcparms) THEN
               BEGIN
               *astr* _ "[can't change archive status of partial
               copy]";
               RETURN( FALSE );
               END;
         END;
   % now do the file %
      % get the old archive status %
         !gtfdb( fjfn, 186+$fdbbkf, $r3);
         arstus _ r3.flarfl.fdbaar;
      % change the archive status %
         IF arstus THEN
            BEGIN
            mask.flarfl.fdbnar _ FALSE;
            mask.flarfl.fdbarc _ FALSE;
            IF astr.L THEN *astr* _ "file and ", *astr*
            ELSE *astr* _ "file already archived";
```

```
            END                                                          02852
         ELSE                                                            02853
            BEGIN                                                        02854
            mask.flarfl.fdbnar _ TRUE;                                   02855
            mask.flarfl.fdbarc _ TRUE;                                   02856
            IF astr.L THEN *astr* _ *astr*, "; archive status changed
            on original file only";                                      03075
            END;                                                         02857
         IF NOT chnfdb( fjfn, $fdbbkf, mask, arcparms) THEN              02722
            BEGIN                                                        02723
            *astr* _ "can't change archive status of the file";          02724
            % reset the pc archive status if changed above %             02725
               IF pcjfn THEN                                             02726
                  BEGIN                                                  02727
                  IF NOT chnfdb( pcjfn, fdbbkf, -1, pcarc) THEN          02728
                     *astr* _ "file archive status not changed;
                     partial copy archive status changed";               02729
                  END;                                                   02730
            RETURN( FALSE );                                             02731
            END;                                                         02732
   % all done, so return %                                               02746
      IF astr.L THEN RETURN(FALSE);                                      03076
      RETURN( TRUE );                                                    02747
                                                                         02748
   END.

                                                                         02749
   %  %                                                                 02750
```

```
(movflandpc) % do the move file stuff %                                 02403
   PROCEDURE                                                            02404
      (jfn1, % jfn for source file %                                    02405
      pcjfn1,        % jfn for source file pc %                        02406
      jfn2,  % jfn for destination file %                               02407
      pcjfn2,        % jfn for destination file pc %                   02408
      errstring);    % address of string to get error messages %
                                                                        02409
   LOCAL movmode, fjfn, fpcjfn;                                         02410
   LOCAL STRING filnam[200], pcnam[200];                                02411
   REF errstring;                                                       02412
                                                                        02413
   % initialize %                                                       02414
      movmode _ TRUE;                                                   02415
   % get local jfns since rnamf will release our group jfn %           02416
      jfnstr( jfn1, $filnam);                                           02417
      IF NOT fjfn _ sgtjfn( gtjoif, $filnam, &errstring) THEN           02418
         BEGIN                                                          02419
         movmode _ FALSE;                                               02420
         *errstring* _ "Can't GTJFN for the file";                      02421
         END;                                                           02422
      IF pcjfn1 THEN                                                    02423
         BEGIN                                                          02424
         jfnstr( pcjfn1, $pcnam);                                       02425
         IF NOT fpcjfn _ sgtjfn( gtjoif, $pcnam, &errstring) THEN
                                                                        02426
            BEGIN                                                       02427
            movmode _ FALSE;                                            02428
            *errstring* _ "[Can't GTJFN for the PC]";                   02429
            END;                                                        02430
         END                                                            02431
      ELSE fpcjfn _ FALSE;                                              02432
   IF movmode THEN                                                      02433
      BEGIN                                                             02434
      % move partial copy first if it exists %                          02435
         IF fpcjfn THEN                                                 02436
            BEGIN                                                       02858
            % propagate protection to new file %                        02442
               proprot( pcjfn1, pcjfn2);                                02443
            IF NOT SKIP !rnamf( fpcjfn, pcjfn2 ) THEN                   02437
               BEGIN                                                    02438
               *errstring* _ "[can't move partial copy]";               02439
               movmode _ FALSE;                                         02440
               END;                                                     02441
            END;                                                        02859
      % now move the file itself %                                      02444
         IF movmode THEN                                                02445
            BEGIN                                                       03188
            % propagate protection to new file %                        02455
               proprot( jfn1, jfn2);                                    02456
            IF NOT SKIP !rnamf( fjfn, jfn2 ) THEN                       02446
               BEGIN                                                    02447
               movmode _ FALSE;                                         02448
               *errstring* _ "can't move this file";                    02449
               % now unmove partial copy if needed %                    02450
                  IF fpcjfn THEN                                        02451
```

```
                    IF NOT SKIP !rnamf( pcjfn2, pcjfn1 ) THEN   02452
                       err( $"system screwup" );                02453
                END;                                            02454
            END;                                                03189
    END;                                                        02457
% get rid of extraneous jfns %                                  02458
    IF NOT SKIP !rljfn(jfn2) THEN NULL;                         02459
    IF NOT SKIP !rljfn(fjfn) THEN NULL;                         02460
    IF pcjfn2 THEN                                              02461
       BEGIN                                                    02462
       IF NOT SKIP !rljfn(fpcjfn) THEN NULL;                    02463
       IF NOT SKIP !rljfn(pcjfn2) THEN NULL;                    02464
       END;                                                     02465
% we're all done now so return %                               02466
    RETURN( movmode );                                          02467
END.
                                                                02468
%  %                                                           02469
```

```
(copflandpc) % do the Copy file stuff %                                  02470
   PROCEDURE                                                             02471
      (jfn1, % jfn for source file %                                     02472
      pcjfn1,        % jfn for source file pc %                         02473
      jfn2,  % jfn for destination file %                               02474
      pcjfn2,        % jfn for destination file pc %                     02475
      errstring);    % address of string to get error messages %
                                                                         02476
   LOCAL copmode;                                                        02477
   LOCAL STRING filnam[200], pcnam[200];                                 02478
   REF errstring;                                                        02479
                                                                         02480
   % initialize %                                                        02481
      copmode _ TRUE;                                                    02482
   % copy partial copy first if it exists %                             02483
      IF pcjfn1 THEN                                                     02484
         IF NOT cflapc( pcjfn1, pcjfn2 ) THEN                            02485
            BEGIN                                                        02486
            *errstring* _ "[can't copy partial copy]";                   02487
            copmode _ FALSE;                                             02488
            END;                                                         02489
   % now copy the file itself %                                         02490
      IF copmode THEN                                                    02491
         IF NOT cflapc( jfn1, jfn2 ) THEN                                02492
            BEGIN                                                        02493
            copmode _ FALSE;                                             02494
            *errstring* _ "can't copy this file";                        02495
            % now delete copied partial copy if needed %                02496
               IF pcjfn1 THEN                                            02497
                  IF NOT SKIP !delf( pcjfn2 ) THEN                       02498
                     err( $"system screwup" );                           02499
            END;                                                         02500
   % get rid of extraneous jfns %                                       02501
      IF NOT SKIP !rljfn(jfn2) THEN NULL;                                02502
      IF pcjfn2 THEN IF NOT SKIP !rljfn(pcjfn2) THEN NULL;               02503
   % we're all done now so return %                                     02504
      RETURN( copmode );                                                 02505
   END.
                                                                         02506
   %  %                                                                 02507
```

```
(proprot)    % propagate protection from jfn1 to jfn2 %                    02508
   PROCEDURE( jfn1, jfn2);                                                 02509
   % get protection of old file %                                          02510
      !gtfdb( jfn1, 1B6 + $fdbprt, $r3);                                   02511
   % now set protection for new file %                                     02512
      IF r3.LH = 500000B THEN chnfdb( jfn2, $fdbprt, 18M, r3);             02513
   RETURN;                                                                 02514
   END.
                                                                           02515
   %  %                                                                    02516
```

```
(cflapc)     % copy file from jfn1 to jfn2 %                                02775
   PROCEDURE( jfn1, jfn2);                                                  02776
   LOCAL page, bytsiz, filsiz, mask;                                        02777
                                                                            02778
   % initialize %                                                           02779
      page _ -1;                                                            02780
   % open the source file %                                                 02781
      IF NOT sysopen( jfn1, read, bintyp, $lit) THEN RETURN( FALSE );
                                                                            02782
   % get byte size and length of source file %                             02783
      !gtfdb( jfn1, 1B6 + $fdbbyv, $bytsiz);                                02784
         mask _ 0;                                                          02785
         bytsiz _ mask.flbyts _ bytsiz.flbyts;                              02786
      !gtfdb( jfn1, 1B6 + $fdbsiz, $filsiz);                                02787
   % open the destination file %                                           02788
      IF NOT SKIP !openf( jfn2, (bytsiz * 1B10) .V 1B5) THEN                02789
         BEGIN                                                              02790
         IF NOT SKIP !closf(4B11 + jfn1) THEN NULL;                         02791
         RETURN( FALSE );                                                   02792
         END;                                                               02793
   % set the size of the destination file %                                02794
      IF NOT chnfdb( jfn2, $fdbsiz, 36M, filsiz) THEN NULL;                 02795
      bytsiz _ 0;  bytsiz.flbyts _ mask.flbyts;                             02796
      mask.flbyts _ -1;                                                     02797
      IF NOT chnfdb( jfn2, $fdbbyv, mask, bytsiz) THEN NULL;                02798
   % loop to get all pages copied %                                        02799
      LOOP                                                                  02800
         BEGIN                                                              02801
         % find used file page %                                            02802
            r1.LH _ jfn1;                                                   02803
            r1.RH _ page + 1;                                               02804
            IF NOT SKIP !ffufp( r1 ) THEN EXIT LOOP;                        02805
            page _ r1.RH;                                                   02806
         % map in this page %                                               02807
            % r1.LH _ jfn1;  r1.RH _ page;  r1 still setup from ffufp
            %                                                               02808
            r2.LH _ 4B5;  r2.RH _ $ckpag/1000B;                             02809
            r3 _ 1B11;        % read access %                               02810
            !pmap( r1, r2, r3);                                             02811
         % copy the page %                                                  02812
            r1.LH _ $ckpag;  r1.RH _ $sfbuff;                               02813
            r2 _ $sfbuff+511;                                               02814
            !BLT r1,(r2);                                                   02815
         % unmap the source page %                                          02816
            r1 _ -1;                                                        02817
            r2.LH _ 4B5;  r2.RH _ $ckpag/1000B;                             02818
            r3 _ 0;                                                         02819
            !pmap( r1, r2, r3);                                             02820
         % map destination page to destination file %                       02821
            r1.LH _ 4B5;  r1.RH _ $sfbuff/1000B;                            02822
            r2.LH _ jfn2;  r2.RH _ page;                                    02823
            r3 _ 1B11;                                                      02824
            !pmap( r1, r2, r3);                                             02825
         % unmap the destination page %                                     02826
            r1 _ -1;                                                        02827
            r2.LH _ 4B5;  r2.RH _ $sfbuff/1000B;                            02828
```

```
            r3 _ 0;                                                    02829
            !pmap( r1, r2, r3);                                        02830
         END;                                                          02831
   % propagate protection to new file %                                02832
      proprot( jfn1, jfn2);                                           02833
   % close the files (don't release the jfns) %                       02834
      IF NOT !closf( 4B11 + jfn1 ) THEN NULL;                         02835
      IF NOT !closf( 4B11 + jfn2 ) THEN NULL;                         02836
   RETURN( TRUE );                                                    02837
   END.
                                                                      02838
   %  %                                                              02839
```

```
(gtdesjfn)   % get destination jfns  ! Can't get JFN if there is
already a deleted file ! %                                          03077
   PROCEDURE                                                        03078
      (jfn1, % source file jfn %                                    03079
      pcjfn1,        % source file pc jfn %                        03080
      udir2, % destination directory name %                         03081
      ufil2, % destination file name %                              03082
      uext2, % destination extension name %                         03083
      uver2, % destination version number %                         03084
      ulft2, % destination miscellaneous stuff %                    03085
      flag2, % destination star field flag bits %                   03086
      errstring      % error string address %                       03087
      );                                                            03088
   LOCAL oldver, dver, jfn2, pcjfn2, temp, flag1, mask, value;      03089
   LOCAL TEXT POINTER dp1, dp2, fp1, fp2;                           03090
   LOCAL STRING                                                     03091
      filnam[200], fil2lnk[200], trash[200], uf1[200], ludir2[60],
      udir1[60], ufil1[60], uext1[40], uver1[40], ulft1[40];        03092
   REF udir2, ufil2, uext2, uver2, ulft2, errstring;                03093
                                                                    03094
   % initialize %                                                   03095
      pcjfn2 _ jfn2 _ 0;                                            03096
   % create destination filename for gtjfn %                       03097
      IF NOT flag2.dstar AND NOT (FIND SF(*udir2*) ['<]) THEN
      *ludir2* _ '<, *udir2*, '> ELSE *ludir2* _ *udir2*;           03233
      IF NOT flag2 THEN                                             03098
         BEGIN                                                      03099
         *filnam* _ *ludir2*, *ufil2*, '., *uext2*, fvrdchar,
         *uver2*;                                                   03100
         IF ulft2.L THEN *filnam* _ *filnam*, ';, *ulft2*;          03101
         END                                                        03102
      ELSE                                                          03103
         BEGIN                                                      03104
         unprseinpt( jfn1, 0, $udir1, $ufil1, $uext1, $uver1, $ulft1,
         $uf1, $flag1);                                             03105
         CASE flag2.dstar OF                                        03107
            = FALSE: *filnam* _ *ludir2* ;                          03108
            ENDCASE *filnam* _ *udir1* ;                            03109
         CASE flag2.fstar OF                                        03110
            = FALSE: *filnam* _ *filnam*, *ufil2*, '.;              03111
            ENDCASE *filnam* _ *filnam*, *ufil1*, '.;               03112
         CASE flag2.estar OF                                        03113
            = FALSE: *filnam* _ *filnam*, *uext2*, fvrdchar;        03114
            ENDCASE *filnam* _ *filnam*, *uext1*, fvrdchar;         03115
         CASE flag2.vstar OF                                        03116
            = FALSE: *filnam* _ *filnam*, *uver2*;                  03117
            ENDCASE *filnam* _ *filnam*, *uver1*;                   03118
         CASE ulft2.L OF                                            03119
            = 0: NULL;                                              03120
            = 1:                                                    03121
               CASE *ulft2*[1] OF                                   03122
                  = '*: *filnam* _ *filnam*, ';, *ulft1*;           03123
                  ENDCASE *filnam* _ *filnam*, ';, *ulft2*;         03124
            ENDCASE *filnam* _ *filnam*, ';, *ulft2*;               03125
         END;                                                       03126
   % create link name for error reporting %                         03127
```

```
    FIND SF(*filnam*) ['<] ^dp1 ['>] ^fp1 ^dp2 _ dp2 SE(*filnam*)
    ^fp2;                                                          03128
    *fil2lnk* _ "< ", + dp1 dp2, ", ", + fp1 fp2, ", >";           03129
  % gtjfn for destination file %                                   03130
    jfn2 _ sgtjfn(gtjidl, $filnam, &errstring);                    03131
    IF jfn2 THEN                                                   03132
       BEGIN                                                       03133
       !gtfdb(jfn2,1B6+$fdbctl, $r3);                              03134
       IF NOT (r3.fdbnxf OR r3.fdbdel) THEN                        03135
          BEGIN                                                    03136
          IF NOT SKIP !rljfn(jfn2) THEN NULL;                      03137
          *errstring* _ *fil2lnk*, " already exists.";             03138
          RETURN(FALSE, FALSE);                                    03139
          END;                                                     03140
       END                                                         03141
    ELSE                                                           03142
       BEGIN                                                       03143
       *errstring* _ "can't get a JFN for ", *fil2lnk*;            03144
       RETURN(FALSE, FALSE);                                       03145
       END;                                                        03146
  % gtjfn for PC for destination if source has a PC %              03147
    IF NOT pcjfn1 THEN RETURN( jfn2, 0);                           03148
    % try to lock the destination file %                          03152
       mask _ value _ 0;                                           03153
       mask.lkinit _ mask.lkdirn _ 36M;                            03154
       value.lkinit _ cinit;                                       03155
       value.lkdirn _ drcode(logdirno);                            03156
       IF NOT chnfdb( jfn2, 24B, mask, value) THEN                 03157
          BEGIN                                                    03158
          IF NOT !rljfn(jfn2 := 0) THEN NULL;                      03159
          *errstring* _ "[can't lock ", *fil2lnk*, '];             03160
          RETURN( FALSE, FALSE);                                   03161
          END;                                                     03162
    % get address of string containing name for new PC %           03163
       temp _ cpcnam( $filnam, logdirno);                          03164
    % get jfn for new PC %                                        03165
       pcjfn2 _ sgtjfn( gtjnfo+gtjidl, temp, &errstring);          03166
    % free the string gotten in cpcnam %                          03167
       freestring( temp, $dspblk);                                 03168
    % report any errors %                                         03169
       IF NOT pcjfn2 THEN                                          03170
          BEGIN                                                    03171
          IF NOT !rljfn(jfn2 := 0) THEN NULL;                      03172
          *errstring* _ "[Can't get a JFN for partial copy for ",
          *fil2lnk*, '];                                           03173
          RETURN( FALSE, FALSE);                                   03174
          END;                                                     03175
  % all done, so return %                                         03176
    RETURN( jfn2, pcjfn2);                                         03177
END.
                                                                   03178
%  %                                                              03179
```

```
%.....copy and show directory utility subroutines....%                    0670
   (didriver)   % main procedure for directory commands %                 01789
      PROCEDURE                                                           01790
         (info, % record desribing what information requested %           01791
         gropk, % record desribing how to group things %                  01792
         sortk, % record describing how to sort things %                  01793
         fjfn,          % driving jfn %                                   01794
         jfnflgs,       % leht half flags for gnjfn %                     01795
         erstrng,       % string for error messages %                     01796
         adlmb  % address of first directory list master block %          01797
         );                                                               01798
      LOCAL                                                               01799
         tptr,          % temp pointer %                                  01800
         pcjfn, % jfn of the pc if it exists %                            01801
         delbit,        % deleted, undeleted, or all type request %
                                                                          01802
         nlschn,        % chain pointer to files w/ pc for 2nd pass %
                                                                          01803
         pcchn, % chain pointer to pc files for 2nd pass %                01804
         chns,  % contains address of data structure chain start %        01805
         adlfb  % address of directory list file blocks %                 01806
         ;                                                                01807
      REF tptr, erstrng, nlschn, pcchn, adlfb, adlmb;                     01808
                                                                          01809
      % initialize locals %                                              01810
         &pcchn _ &nlschn _ chns _ &adlfb _ 0;                            01811
         delbit _ info.dlidlt := FALSE;                                   01812
      % turn off the FOR FILE field in the info record %                 01813
         info.dlifrf _ FALSE;                                             01814
      % loop to get all files %                                          01815
         LOOP                                                             01816
            BEGIN                                                         01817
            % exit if control-O interrupt %                               02664
               IF inptrf THEN EXIT LOOP;                                  02665
            % see if this file has proper deletion status %              01818
               IF NOT dlckdl( fjfn, delbit) THEN                          01819
                  IF NOT SKIP !gnjfn(jfnflgs * 1B6 + fjfn) THEN           01820
                     BEGIN                                                01821
                     % dont return null chain in this case %              01822
                        IF NOT chns THEN chns _ dlgtblk( &adlmb, dlgbl);
                                                                          01823
                     EXIT LOOP;                                           01824
                     END                                                  01825
                  ELSE REPEAT LOOP;                                       01826
            % get pc jfn if it exists %                                  01827
               erstrng.L _ 0;                                             01828
               getpcjfn( fjfn, $pcjfn, $erstrng, TRUE);                   01829
            % get directory list file block for this file (and pc) %
                                                                          01830
               IF NOT                                                     01831
                  (&adlfb _ mkdirlist(fjfn, pcjfn, info, gropk, sortk,
                  &adlmb))                                                01832
                  THEN                                                    01833
                     BEGIN                                                01834
                     *erstrng* _ "Ran Out Of Space Before Finishing";
                                                                          01835
```

```
                % get rid of redundant pc files %                    01836
                   dlsnps( &nlschn, &pcchn);                         01837
                RETURN( chns );                                      01838
                END;                                                 01839
          % add this dlfb to the data structure %                    01840
            IF NOT (dlinfb($chns, &adlmb, &adlfb, sortk)) THEN       01841
               BEGIN                                                 01842
               *erstrng* _ "Ran Out Of Space Before Finishing";      01843
               % get rid of redundant pc files %                     01844
                  dlsnps( &nlschn, &pcchn);                          01845
               RETURN( chns );                                       01846
               END;                                                  01847
          % handle any errors from getpcjfn above %                  01848
            IF erstrng.L THEN                                        01849
               IF ( &tptr _                                          01850
               dlgtblk( &adlmb, [adlfb.dlfbal].L+erstrng.L+1+2) )
                                                                     01851
               THEN                                                  01852
                  BEGIN                                              01853
                  tptr.M _ [adlfb.dlfbal].L + erstrng.L + 4;         01854
                  *tptr* _ *[adlfb.dlfbal]*, "  [", *erstrng*, "];
                                                                     01855
                  adlfb.dlfbal _ &tptr;                              01856
                  END                                                01857
               ELSE                                                  01858
                  BEGIN                                              01859
                  *erstrng* _ "Ran Out Of Space Before Finishing";
                                                                     01860
                  % get rid of redundant pc files %                  01861
                     dlsnps( &nlschn, &pcchn);                       01862
                  RETURN( chns );                                    01863
                  END;                                               01864
          % create second pass chains %                              01865
            IF adlfb.dlfbfl.dlstpc THEN                              01866
               BEGIN                                                 01867
               adlfb.dlfbcp _ &pcchn;                                01868
               &pcchn _ &adlfb;                                      01869
               END                                                   01870
            ELSE IF adlfb.dlfbfl.dlstnl THEN                         01871
               BEGIN                                                 01872
               adlfb.dlfbcp _ &nlschn;                               01873
               &nlschn _ &adlfb;                                     01874
               END;                                                  01875
          % go on to the next file %                                 01876
            IF pcjfn THEN IF NOT SKIP !rljfn( pcjfn ) THEN NULL;
                                                                     01877
            IF NOT SKIP !gnjfn( jfnflgs * 1B6 + fjfn ) THEN EXIT
            LOOP;                                                    01878
          END;                                                       01879
      % get rid of redundant pc files %                             01880
        dlsnps( &nlschn, &pcchn);                                    01881
      % i think we're done so i'll return %                         01882
        RETURN( chns );                                              01883
                                                                     01884
      END.
                                                                     01885
```

% %

01886

```
(dlgmb)      % get and initialize a directory list master block %
                                                                    01927
   PROCEDURE                                                        01928
      (blksiz);      % size of desired block %                     01929
   LOCAL                                                            01930
      temp,          % temp for use in allocating file pages %     01968
      block; % address of allocated block %                        01931
   REF block;                                                       01932
                                                                    01933
   % FUNCTION %                                                    01934
      % this procedure gets and set up a directory list master block
      of core that can be used to allcate the needed blocks for the
      directory (and similar) commands. %                           01935
   % RETURNS %                                                     01936
      % returns the address of the allocated master block on
      succesful completion or FALSE on unsuccesfull completion %
                                                                    01937
                                                                    01938
   % get a block (or file pages) and initialize the storage %      01939
      IF NOT (&block _ getblk( blksiz, $dspblk)) THEN               01940
         BEGIN       % ran out of block storage: use file pages %
                                                                    01941
         IF NOT oldpgsz THEN temp _ upgbsz;                         01942
         IF NOT &block _                                            01943
            gpbsz( MAX(upgbsz+2, upgbsz+((blksiz+511)/512)+1 ) )
                                                                    01965
            THEN RETURN( FALSE );                                   01966
         IF NOT oldpgsz THEN oldpgsz _ temp;                        01967
         % zero the new pages %                                     01944
            block _ 0;                                              01945
            r1.LH _ &block;                                         01946
            r1.RH _ &block + 1;                                     01947
            !BLT r1,upgbend;                                        01948
         % now set it up to look like regular block %               01949
            block.dlmbln _ block.dlmbal _ upgbend-&block-dlmbl-1;
                                                                    01950
            block.dlmbfp _ &block + dlmbl;                          01951
         END                                                        01952
      ELSE                                                          01953
         BEGIN                                                      01954
         % now initialize the block %                               01955
            block.dlmbzn _ $dspblk;                                 01956
            block.dlmbnm _ 0;                                       01957
            block.dlmbfp _ &block + dlmbl;                          01958
         END;                                                       01959
   % now return %                                                  01960
      RETURN( &block );                                             01961
                                                                    01962
   END.
                                                                    01963
   %  %                                                           01964
```

```
(dlfmb)      % free directory list master blocks %                    01903
   PROCEDURE                                                          01904
      (ablk  % address of first master block %                        01905
      );                                                              01906
   LOCAL                                                              01907
      temp;                                                           01908
   REF ablk;                                                          01909
                                                                      01910
   % ALGORITHM %                                                      01911
      % frees all master blocks in one chain starting with the master
      block whose address is passed %                                 01912
                                                                      01913
   % free all blocks in the chain and return %                        01914
      WHILE &ablk DO                                                  01915
         BEGIN                                                        01916
         temp _ ablk.dlmbnm;                                          01917
         IF ablk.dlmbzn THEN freeblk( &ablk, ablk.dlmbzn );           01918
         &ablk _ temp;                                                01919
         END;                                                         01920
      % now free any file pages %                                     01921
         IF oldpgsz THEN gpbsz(oldpgsz := 0);                         01922
      RETURN;                                                         01923
                                                                      01924
   END.
                                                                      01925
   %  %                                                               01926
```

```
(dlgtblk)    % get a directory block of length blksiz %                    0810
   PROCEDURE                                                               0811
      (ablk,  % master block address within which to allocate block %
                                                                           0812
      blksiz  % desired block size %                                       0813
      );                                                                   0814
   LOCAL                                                                   0815
      temp;                                                                0816
   REF ablk;                                                               0817
                                                                           0818
   % RETURNS %                                                            0819
      % returns the address of the allocated storage block or FALSE %
                                                                           0820
                                                                           0821
   % if room for new block, update free pointer and return %              0822
      LOOP                                                                 0823
         IF ((ablk.dlmbfp - &ablk + blksiz) < ablk.dlmbln)                 0824
            THEN RETURN ( ablk.dlmbfp := ablk.dlmbfp + blksiz )            0825
         ELSE                                                              0826
            IF ablk.dlmbnm # 0 THEN &ablk _ ablk.dlmbnm                    0827
            ELSE                                                           0828
               BEGIN                                                       0829
               IF NOT ( temp_dlgmb(MAX(ablk.dlmbln, blksiz+dlmbl+1))
               )                                                           0830
                  THEN IF NOT ( temp_dlgmb(blksiz+dlmbl+1) ) THEN
                                                                           0831
                     RETURN( FALSE );                                      0832
               &ablk _ ablk.dlmbnm _ temp;                                 0833
               END;                                                        0834
                                                                           0835
   END.
                                                                           0836
   %  %                                                                 01888
```

```
(mkdirlist)  % make an entry for a directory list %                    0837
   PROCEDURE                                                           0838
      (fjfn, % jfn of file to be listed %                              0839
      pcjfn, % jfn of partial copy for file if it exists %             0840
      info,  % record (dlinfo) for which fields are to be listed %
                                                                       0841
      gropk, % record (dlgrp) for what grouping is to be done %        0842
      sortk, % record (dlsort) for what sorting is to be done %        0843
      adlmb  % address of first directory list master block %          0844
      );                                                               0845
   LOCAL                                                               0846
      adlfb, adfdb, adfdbpc; % address of a directory list file block
      %                                                                0847
   LOCAL STRING                                                        0848
      filelink[200], % temp string for file link this file %           0849
      pcname[200]    % temp string for pc name or file name %          0850
      ;                                                                0851
   REF adlmb, adlfb, adfdb, adfdbpc;                                   0852
                                                                       0853
   % RETURNS %                                                        0854
      % This routine returns the address of a directory list file
      block (obtained via dlgtblk and described by the record dlflbk)
      or FALSE if it fails because of running out of space %           0855
                                                                       0856
   % get a directory list file block %                                0857
      IF NOT (&adlfb _ dlgtblk( &adlmb, dlfbl)) THEN RETURN( FALSE );
                                                                       0858
   % get fdb for the file (and account if neccessary) %               0859
      IF NOT (adlfb.dlfbff _ &adfdb _ dlgtblk( &adlmb, 26B)) THEN
                                                                       0860
         RETURN( FALSE );                                              0861
      !gtfdb( fjfn, 25000000B, adlfb.dlfbff);                          0862
      IF t2v3flag THEN                                                03192
         BEGIN                                                        03193
         adfdb.xfdbjfn _ fjfn;                                        03195
         END;                                                         03194
      IF info.dliacc THEN dlbgac(fjfn, adlfb.dlfbff, &adlmb);          0863
   % get fdb for pc if neccessary (and account if neccessary)%        0864
      IF pcjfn THEN                                                    0865
         BEGIN                                                         0866
         IF NOT (adlfb.dlfbpf _ &adfdbpc _ dlgtblk( &adlmb, 26B))
         THEN                                                          0867
            RETURN( FALSE );                                           0868
         !gtfdb( pcjfn, 25000000B, adlfb.dlfbpf);                      0869
         IF t2v3flag THEN                                             03196
            BEGIN                                                     03197
            adfdbpc.xfdbjfn _ pcjfn;                                  03198
            END;                                                      03199
         IF info.dliacc THEN dlbgac(pcjfn, adlfb.dlfbpf, &adlmb);
                                                                       0870
         END;                                                          0871
   % build the file link for this file %                              0872
      jfnflink( fjfn, $filelink, 011110B6+1);                          0873
   % build string for second pass ( and set bits in dlfb %            0874
      dlblds( fjfn, pcjfn, $pcname, &adlfb);                           0875
   % get the group key for this file %                                0876
```

```
    adlfb.dlfbgk _ dlgetg( gropk, adlfb.dlfbff, pcjfn,
    adlfb.dlfbpf);                                                  0877
  % get the sort key for this file %                                0878
    adlfb.dlfbsk _ dlgets( sortk, adlfb.dlfbff, pcjfn,
    adlfb.dlfbpf);                                                  0879
  % get directory list storage for the file link %                  0880
    IF (adlfb.dlfbal _ dlgtblk( &adlmb, (filelink.L+4)/5+1 )) THEN
                                                                    0881
       BEGIN                                                        0882
       [adlfb.dlfbal].M _ filelink.L;                               0883
       % place filelink in directory list storage %                0884
          *[adlfb.dlfbal]* _ *filelink*;                            0885
       END                                                          0886
    ELSE RETURN( FALSE );                                           0887
  % get directory list storage for second pass string if needed %
                                                                    0888
    IF pcname.L THEN                                                0889
       IF (adlfb.dlfbap _ dlgtblk( &adlmb, (pcname.L+4)/5+1 )) THEN
                                                                    0890
          BEGIN                                                     0891
          [adlfb.dlfbap].M _ pcname.L;                              0892
          % place second pass string in directory list storage %
                                                                    0893
             *[adlfb.dlfbap]* _ *pcname*;                           0894
          END                                                       0895
       ELSE RETURN( FALSE );                                        0896
    IF adlfb.dlfbfl.dlstpc THEN adlfb.dlfbap _ adlfb.dlfbal;        0897
  % all done, so return address of the directory list file block %
                                                                    0898
    RETURN( &adlfb );                                               0899
                                                                    0900
END.
                                                                    0901
%  %                                                              01890
```

```
(dlbldai)    % build additional line 1 information string %          0902
   PROCEDURE                                                         0903
      (fjfn, % jfn for the file %                                    0904
      pcjfn, % jfn of the pc if it exists %                          0905
      jfnfdb,        % address of fdb for the file %                 0906
      pcfdb, % address of the fdb for the pc %                       0907
      string % address of string to receive the information %        0908
      );                                                             0909
   LOCAL STRING                                                      0910
      locstr[100];   % temp string %                                 0911
   REF jfnfdb, pcfdb, string;                                        0912
                                                                     0913
   % get scratch or temporary info %                                0914
      IF jfnfdb[$fdbctl].fdbtmp THEN                                 0915
         IF jfnfdb[$fdbver].LH >= 100000 THEN                        0916
            *string* _ *string*, "  Scratch"                         0917
         ELSE *string* _ *string*, "  Temporary";                   0918
   % get ephemeral status %                                         0919
      IF jfnfdb[$fdbctl].fdbeph THEN                                 0920
         *string* _ *string*, "  Ephemeron";                         0921
   % get deletion status %                                          0922
      IF jfnfdb[$fdbctl].fdbdel THEN                                 0923
         *string* _ *string*, "  Deleted";                           0924
   % now handle the pc %                                            0925
      IF pcjfn THEN                                                  0926
         BEGIN                                                       0927
         lockid( $locstr, jfnfdb[$fdbusw].lkdirnum,                  0928
            jfnfdb[$fdbusw].lkinit );                                0929
         *string* _ *string*, "  [", *locstr*, SP;                   0930
         % get scratch or temporary info %                           0931
            IF pcfdb[$fdbctl].fdbtmp THEN                            0932
               IF pcfdb[$fdbver].LH >= 100000 THEN                   0933
                  *string* _ *string*, "Scratch "                    0934
               ELSE *string* _ *string*, "Temporary ";               0935
         % get ephemeral status %                                    0936
            IF pcfdb[$fdbctl].fdbeph THEN                            0937
               *string* _ *string*, "Ephemeron ";                    0938
         % get deletion status %                                     0939
            IF pcfdb[$fdbctl].fdbdel THEN                            0940
               *string* _ *string*, "Deleted";                       0941
         *string* _ *string*, ']';                                   0942
         END;                                                        0943
   % done so return %                                               0944
      RETURN;                                                        0945
                                                                     0946
   END.                                                              0947
   %  %                                                            01891
```

```
(dlbldi)     % build information string for this file %                    0948
   PROCEDURE                                                              0949
      (info, % dlinfo record for what is to be listed %                   0950
      pcjfn, % TRUE if pc exists %                                        0951
      jfnfdb,        % address of fdb for the file %                       0952
      pcfdb, % address of the fdb for the pc %                            0953
      lead,  % address of strng to precede each line %                    0954
      string % address of string to receive the information %             0955
      );                                                                  0956
   REF jfnfdb, pcfdb, lead, string;                                       0957
                                                                          0958
   % quick return if nothing requested %                                  0959
      IF NOT info THEN RETURN;                                            0960
   % get the account of this file %                                       0961
      IF info.dliacc THEN                                                 0962
         BEGIN                                                            0963
         *string* _ *string*, *lead*, "Account: ";                        0964
         CASE jfnfdb[$fdbact] OF                                          0965
            = -1: *string* _ *string*, "Can't Get File Account";          0966
            >  0: *string* _ *string*, STRING( jfnfdb[$fdbact] );
                                                                          0967
            <  0: *string* _ *string*, *[jfnfdb[$fdbact].RH]*;            0968
            ENDCASE;                                                      0969
         IF pcjfn THEN                                                    0970
            BEGIN                                                         0971
            *string* _ *string*, "   [";                                  0972
            CASE pcfdb[$fdbact] OF                                        0973
               = -1: *string* _ *string*, "Can't Get File Account";
                                                                          0974
               >  0: *string* _ *string*, STRING( pcfdb[$fdbact] );
                                                                          0975
               <  0: *string* _ *string*, *[pcfdb[$fdbact].RH]*;          0976
               ENDCASE;                                                   0977
            *string* _ *string*, "]";                                     0978
            END;                                                          0979
         *string* _ *string*, CR, LF;                                     0980
         END;                                                             0981
   % get the archive status of this file %                                0982
      IF info.dliars THEN                                                 0983
         BEGIN                                                            0984
         *string* _ *string*, *lead*, "Archive Status: ";                 0985
         dlbgas( jfnfdb[$fdbbkf].flarfl, &string);                        0986
         IF pcjfn THEN                                                    0987
            BEGIN                                                         0988
            *string* _ *string*, CR, LF, *lead*, "   [";                 0989
            dlbgas( pcfdb[$fdbbkf].flarfl, &string);                      0990
            *string* _ *string*, "]";                                     0991
            END;                                                          0992
         *string* _ *string*, CR, LF;                                     0993
         END;                                                             0994
   % get the archive tape numbers of this file %                          0995
      IF info.dliart THEN                                                 0996
         BEGIN                                                            0997
         *string* _ *string*, *lead*, "Archive Tapes: ";                  0998
         IF NOT jfnfdb[$fdbart] THEN                                      0999
            IF jfnfdb[$fdbbkf].flarfl.fdbaar THEN                        01000
```

```
          *string* _ *string*, "Not Available"                01001
       ELSE *string* _ *string*, "Not Archived"               01002
    ELSE                                                      01003
       *string* _ *string*, STRING( jfnfdb[$fdbart].flfsat), "
       and ", STRING( jfnfdb[$fdbart].flsnat);                01004
    IF pcjfn THEN                                             01005
       BEGIN                                                  01006
       *string* _ *string*, "   [";                           01007
       IF NOT pcfdb[$fdbart] THEN                             01008
          IF pcfdb[$fdbbkf].flarfl.fdbaar THEN                01009
             *string* _ *string*, "Not Available"             01010
          ELSE *string* _ *string*, "Not Archived"            01011
       ELSE                                                   01012
          *string* _ *string*, STRING( pcfdb[$fdbart].flfsat), "
          and ", STRING( pcfdb[$fdbart].flsnat);              01013
       *string* _ *string*, '];                               01014
       END;                                                   01015
    *string* _ *string*, CR, LF;                              01016
    END;                                                      01017
% get the dump tape number of this file %                     01018
  IF info.dlidmt THEN                                         01019
    BEGIN                                                     01020
    *string* _ *string*, *lead*, "Dump Tape: ";               01021
    IF jfnfdb[$fdbbkf].fldmpt THEN                            01022
       *string* _ *string*, STRING( jfnfdb[$fdbbkf].fldmpt )
                                                              01023
    ELSE                                                      01024
       IF jfnfdb[$fdbtdm] THEN                                01025
          *string* _ *string*, "Not Available"                01026
       ELSE                                                   01027
          *string* _ *string*, "Not Dumped";                  01028
    IF pcjfn THEN                                             01029
       BEGIN                                                  01030
       *string* _ *string*, "   [";                           01031
       IF pcfdb[$fdbbkf].fldmpt THEN                          01032
          *string* _ *string*, STRING( pcfdb[$fdbbkf].fldmpt )
                                                              01033
       ELSE                                                   01034
          IF pcfdb[$fdbtdm] THEN                              01035
             *string* _ *string*, "Not Available"             01036
          ELSE                                                01037
             *string* _ *string*, "Not Dumped";               01038
       *string* _ *string*, '];                               01039
       END;                                                   01040
    *string* _ *string*, CR, LF;                              01041
    END;                                                      01042
% get the number of versions to keep of this file %           01043
  IF info.dlidfr THEN                                         01044
    BEGIN                                                     01045
    *string* _ *string*, *lead*, "No. Versions To Keep: ",    01046
       STRING( jfnfdb[$fdbbyv].fldfr );                       01047
    IF pcjfn THEN                                             01048
       *string* _ *string*,                                   01049
       "   [", STRING( pcfdb[$fdbbyv].fldfr ), '];            01050
    *string* _ *string*, CR, LF;                              01051
    END;                                                      01052
```

```
   % get the last writer of this file %                                01053
     IF info.dlilwr THEN                                               01054
        BEGIN                                                          01055
        *string* _ *string*, *lead*, "Last Writer: ";                  01056
        dlbglw( jfnfdb[$fdbuse].LH, &string, jfnfdb[$xfdbjfn]);
                                                                       01057
        IF pcjfn THEN                                                  01058
           BEGIN                                                       01059
           *string* _ *string*, "   [";                               01060
           dlbglw( pcfdb[$fdbuse].LH, &string, pcfdb[$xfdbjfn]);
                                                                       01061
           *string* _ *string*, ']';                                   01062
           END;                                                        01063
        *string* _ *string*, CR, LF;                                   01064
        END;                                                           01065
   % get the length and bytesize of this file %                        01066
     IF info.dlibyt THEN                                               01067
        BEGIN                                                          01068
        *string* _ *string*, *lead*, "Length (and Bytesize): ",
                                                                       01069
           STRING( jfnfdb[$fdbsiz] ),                                  01070
           '(, STRING( jfnfdb[$fdbbyv].flbyts ), ');                   01071
        IF pcjfn THEN                                                  01072
           *string* _ *string*, "   [",                               01073
           STRING( pcfdb[$fdbsiz] ),                                   01074
           '(, STRING( pcfdb[$fdbbyv].flbyts ), ")]";                  01075
        *string* _ *string*, CR, LF;                                   01076
        END;                                                           01077
   % get the number of accesses of this file %                         01078
     IF info.dlinrw THEN                                               01079
        BEGIN                                                          01080
        *string* _ *string*, *lead*,                                   01081
           "No. of Accesses (reads + writes): ",                       01082
           STRING( jfnfdb[$fdbcnt].flnmrd + jfnfdb[$fdbcnt].flnmwr
           ),
                                                                       01083
           " (", STRING( jfnfdb[$fdbcnt].flnmrd ), " + ",              01084
           STRING( jfnfdb[$fdbcnt].flnmwr ), ');                       01085
        IF pcjfn THEN                                                  01086
           *string* _ *string*, "   [",                               01087
           STRING( pcfdb[$fdbcnt].flnmrd + pcfdb[$fdbcnt].flnmwr ),
                                                                       01088
           " (", STRING( pcfdb[$fdbcnt].flnmrd ), " + ",               01089
           STRING( pcfdb[$fdbcnt].flnmwr ), ")]";                      01090
        *string* _ *string*, CR, LF;                                   01091
        END;                                                           01092
   % get miscellaneous information for this file %                     01093
     IF info.dlimis THEN                                               01094
        BEGIN                                                          01095
        *string* _ *string*, *lead*, "Misc. Info.: ";                  01096
        IF jfnfdb[$fdbctl].fdblng .V jfnfdb[$fdbctl].fdbprm THEN
                                                                       01097
           BEGIN                                                       01098
           IF jfnfdb[$fdbctl].fdblng THEN                              01099
              *string* _ *string*, "Long File  ";                      01100
           IF jfnfdb[$fdbctl].fdbprm THEN                              01101
              *string* _ *string*, "Permanent File";                   01102
```

```
            END                                                    01103
         ELSE *string* _ *string*, "None";                         01104
         IF pcjfn THEN                                             01105
            BEGIN                                                  01106
            *string* _ *string*, "   [";                          01107
            IF jfnfdb[$fdbctl].fdblng .V jfnfdb[$fdbctl].fdbprm THEN
                                                                   01108
               BEGIN                                               01109
               IF jfnfdb[$fdbctl].fdblng THEN                      01110
                  *string* _ *string*, "Long File  ";              01111
               IF jfnfdb[$fdbctl].fdbprm THEN                      01112
                  *string* _ *string*, "Permanent File";           01113
               END                                                 01114
            ELSE *string* _ *string*, "None";                      01115
            *string* _ *string*, ']';                              01116
            END;                                                   01117
         *string* _ *string*, CR, LF;                              01118
         END;                                                      01119
   % get the protection of this file %                             01120
      IF info.dliprt THEN                                          01121
         BEGIN                                                     01122
         *string* _ *string*, *lead*, "Protection: ";              01123
         IF jfnfdb[$fdbprt].LH # 500000B THEN                      01124
            *string* _ *string*,                                   01125
               '(, STRING( jfnfdb[$fdbprt], 8), ") - Fancy
               Protection"                                         01126
         ELSE                                                      01127
            BEGIN                                                  01128
            *string* _ *string*,                                   01129
               '(, STRING( jfnfdb[$fdbprt].RH, 8), '), CR, LF,     01130
                 *lead*, "  Self ";                               01131
               dlbgpr( jfnfdb[$fdbprt].flprse, &string );          01132
               *string* _ *string*, CR, LF, *lead*, "  Group ";    01133
               dlbgpr( jfnfdb[$fdbprt].flprgr, &string );          01134
               *string* _ *string*, CR, LF, *lead*, "  Public ";
                                                                   01135
               dlbgpr( jfnfdb[$fdbprt].flprpu, &string );          01136
            END;                                                   01137
         IF pcjfn THEN                                             01138
            BEGIN                                                  01139
            *string* _ *string*, CR, LF, *lead*, "   [ ";         01140
            IF pcfdb[$fdbprt].LH # 500000B THEN                    01141
               *string* _ *string*,                                01142
                  '(, STRING( pcfdb[$fdbprt], 8), ") - Fancy
                  Protection ]"                                    01143
            ELSE                                                   01144
               BEGIN                                               01145
               *string* _ *string*,                                01146
                  '(, STRING( pcfdb[$fdbprt].RH, 8), '), CR, LF,
                                                                   01147
                    *lead*, "    Self ";                           01148
                  dlbgpr( pcfdb[$fdbprt].flprse, &string );        01149
                  *string* _ *string*, CR, LF, *lead*, "    Group ";
                                                                   01150
                  dlbgpr( pcfdb[$fdbprt].flprgr, &string );        01151
                  *string* _ *string*, CR, LF, *lead*, "    Public ";
```

```
                                                                    01152
                dlbgpr( pcfdb[$fdbprt].flprpu, &string );           01153
                *string* _ *string*, " ]";                          01154
              END;                                                  01155
           END;                                                     01156
        *string* _ *string*, CR, LF;                               01157
        END;                                                        01158
    % get the size of this file %                                   01159
      IF info.dlisiz THEN                                           01160
        BEGIN                                                       01161
        *string* _ *string*, *lead*, "Size in Pages: ",             01162
           STRING( jfnfdb[$fdbbyv].flpgsz );                        01163
        IF pcjfn THEN                                               01164
           *string* _ *string*,                                     01165
              "   [", STRING( pcfdb[$fdbbyv].flpgsz ), ']';         01166
        *string* _ *string*, CR, LF;                                01167
        END;                                                        01168
    % get [time and] date of archiving of this file %              01169
      IF info.dlitar THEN                                           01170
        BEGIN                                                       01171
        *string* _ *string*, *lead*, "Archived: ";                  01172
        IF jfnfdb[$fdbtsa] THEN                                     01173
           dlbgtd( info.dlitar, jfnfdb[$fdbtsa], &string)           01174
        ELSE                                                        01175
           IF jfnfdb[$fdbbkf].flarfl.fdbaar THEN                    01176
              *string* _ *string*, "Not Available"                  01177
           ELSE                                                     01178
              *string* _ *string*, "Not Archived";                  01179
        IF pcjfn THEN                                               01180
           BEGIN                                                    01181
           *string* _ *string*, "   [";                             01182
           IF pcfdb[$fdbtsa] THEN                                   01183
              dlbgtd( info.dlitar, pcfdb[$fdbtsa], &string)         01184
           ELSE                                                     01185
              IF pcfdb[$fdbbkf].flarfl.fdbaar THEN                  01186
                 *string* _ *string*, "Not Available"               01187
              ELSE                                                  01188
                 *string* _ *string*, "Not Archived";               01189
           *string* _ *string*, ']';                                01190
           END;                                                     01191
        *string* _ *string*, CR, LF;                                01192
        END;                                                        01193
    % get [time and] date of creation of this file %               01194
      IF info.dlitcr THEN                                           01195
        BEGIN                                                       01196
        *string* _ *string*, *lead*, "Created: ";                   01197
        IF jfnfdb[$fdbcrv] THEN                                     01198
           dlbgtd( info.dlitcr, jfnfdb[$fdbcrv], &string)           01199
        ELSE *string* _ *string*, "Not Available";                  01200
        IF pcjfn THEN                                               01201
           BEGIN                                                    01202
           *string* _ *string*, "   [";                             01203
           IF pcfdb[$fdbcrv] THEN                                   01204
              dlbgtd( info.dlitcr, pcfdb[$fdbcrv], &string)         01205
           ELSE *string* _ *string*, "Not Available";               01206
           *string* _ *string*, ']';                                01207
```

```
        END;                                                    01208
      *string* _ *string*, CR, LF;                              01209
      END;                                                      01210
% get [time and] date of dumping of this file %                  01211
   IF info.dlitdm THEN                                          01212
      BEGIN                                                     01213
      *string* _ *string*, *lead*, "Dumped: ";                  01214
      IF jfnfdb[$fdbtdm] THEN                                   01215
         dlbgtd( info.dlitdm, jfnfdb[$fdbtdm], &string)         01216
      ELSE *string* _ *string*, "Not Dumped";                   01217
      IF pcjfn THEN                                             01218
         BEGIN                                                  01219
         *string* _ *string*, "   [";                          01220
         IF pcfdb[$fdbtdm] THEN                                 01221
            dlbgtd( info.dlitdm, pcfdb[$fdbtdm], &string)       01222
         ELSE *string* _ *string*, "Not Dumped";                01223
         *string* _ *string*, ];                                01224
         END;                                                   01225
      *string* _ *string*, CR, LF;                              01226
      END;                                                      01227
% get [time and] date of original version creation of this file %
                                                                01228
   IF info.dlitov THEN                                          01229
      BEGIN                                                     01230
      *string* _ *string*, *lead*, "Original Version Created: ";
                                                                01231
      IF jfnfdb[$fdbcre] THEN                                   01232
         dlbgtd( info.dlitov, jfnfdb[$fdbcre], &string)         01233
      ELSE *string* _ *string*, "Not Available";                01234
      IF pcjfn THEN                                             01235
         BEGIN                                                  01236
         *string* _ *string*, "   [";                          01237
         IF pcfdb[$fdbcre] THEN                                 01238
            dlbgtd( info.dlitov, pcfdb[$fdbcre], &string)       01239
         ELSE *string* _ *string*, "Not Available";             01240
         *string* _ *string*, ];                                01241
         END;                                                   01242
      *string* _ *string*, CR, LF;                              01243
      END;                                                      01244
% get [time and] date of last reading of this file %            01245
   IF info.dlitrd THEN                                          01246
      BEGIN                                                     01247
      *string* _ *string*, *lead*, "Last Read: ";               01248
      IF jfnfdb[$fdbref] THEN                                   01249
         dlbgtd( info.dlitrd, jfnfdb[$fdbref], &string)         01250
      ELSE *string* _ *string*, "Never Read";                   01251
      IF pcjfn THEN                                             01252
         BEGIN                                                  01253
         *string* _ *string*, "   [";                          01254
         IF pcfdb[$fdbref] THEN                                 01255
            dlbgtd( info.dlitrd, pcfdb[$fdbref], &string)       01256
         ELSE *string* _ *string*, "Never Read";                01257
         *string* _ *string*, ];                                01258
         END;                                                   01259
      *string* _ *string*, CR, LF;                              01260
      END;                                                      01261
```

```
   % get [time and] date of last writing of this file %                    01262
      IF info.dlitwr THEN                                                  01263
         BEGIN                                                             01264
         *string* _ *string*, *lead*, "Last Written: ";                    01265
         IF jfnfdb[$fdbwrt] THEN                                           01266
            dlbgtd( info.dlitwr, jfnfdb[$fdbwrt], &string)                 01267
         ELSE *string* _ *string*, "Never Written";                        01268
         IF pcjfn THEN                                                     01269
            BEGIN                                                          01270
            *string* _ *string*, "   [";                                   01271
            IF pcfdb[$fdbwrt] THEN                                         01272
               dlbgtd( info.dlitwr, pcfdb[$fdbwrt], &string)               01273
            ELSE *string* _ *string*, "Never Written";                     01274
            *string* _ *string*, ];                                        01275
            END;                                                           01276
         *string* _ *string*, CR, LF;                                      01277
         END;                                                              01278
   % done finally so return %                                              01279
      RETURN;                                                              01280
                                                                           01281
   END.
                                                                           01282
   %  %                                                                   01892
```

```
(dlgetg)     % return the group key for this file %                    01283
   PROCEDURE                                                          01284
      (gropk,         % dlgrp record %                                 01285
      jfnfdb,         % address of the file fdb %                      01286
      pcjfn, % jfn for the pc if it exists %                          01287
      pcfdb  % address of the pc fdb %                                01288
      );                                                              01289
   REF jfnfdb, pcfdb;                                                 01290
                                                                      01291
   % recurse if reverse grouping %                                    01292
      IF (gropk.dlgrvr := FALSE) THEN                                 01293
         RETURN( dlgetg(gropk, &jfnfdb, pcjfn, &pcfdb) .X -1 );       01294
   % see if any grouping %                                           01295
      IF NOT gropk THEN RETURN( 35M );       % no grouping %         01296
   IF gropk.dlgacc THEN       % group by account %                    01297
      IF jfnfdb[$fdbact] < 0 THEN RETURN( jfnfdb[$fdbact].RH )        01298
      ELSE RETURN( jfnfdb[$fdbact] );                                 01299
   IF gropk.dlgars THEN       % group by archive status %             01300
      RETURN( jfnfdb[$fdbbkf].flarfl );                               01301
   IF gropk.dlgdar THEN       % group by archive date %               01302
      RETURN( jfnfdb[$fdbtsa].LH );                                   01303
   IF gropk.dlgart THEN       % group by archive tapes %              01304
      RETURN( jfnfdb[$fdbart].flsnat );                               01305
   IF gropk.dlgbyt THEN       % group by bytesize %                   01306
      RETURN( jfnfdb[$fdbbyv].flbyts );                               01307
   IF gropk.dlgdcr THEN       % group by creation date %              01308
      RETURN( jfnfdb[$fdbcrv].LH );                                   01309
   IF gropk.dlgdlt THEN       % group by deletion status %            01310
      RETURN( jfnfdb[$fdbctl].fdbdel );                               01311
   IF gropk.dlgddm THEN       % group by dump date %                  01312
      RETURN( jfnfdb[$fdbtdm].LH );                                   01313
   IF gropk.dlgdmt THEN       % group by dump tape %                  01314
      RETURN( jfnfdb[$fdbbkf].fldmpt );                               01315
   IF gropk.dlgdfr THEN       % group by no. of versions to keep %    01316
      RETURN( jfnfdb[$fdbbyv].fldfr );                                01317
   IF gropk.dlglwr THEN       % group by last writer %                01318
      IF pcjfn THEN RETURN( pcfdb[$fdbuse].LH )                       01319
      ELSE RETURN( jfnfdb[$fdbuse].LH );                              01320
   IF gropk.dlgdov THEN       % group by orig. vers. creation date %  01321
      RETURN( jfnfdb[$fdbcre].LH );                                   01322
   IF gropk.dlgprt THEN       % group by protection %                 01323
      RETURN( jfnfdb[$fdbprt] );                                      01324
   IF gropk.dlgdrd THEN       % group by last read date %             01325
      IF pcjfn THEN                                                   01326
         RETURN( MAX( jfnfdb[$fdbref].LH, pcfdb[$fdbref].LH) )        01327
      ELSE RETURN( jfnfdb[$fdbref].LH );                              01328
   IF gropk.dlgdwr THEN       % group by last write date %            01329
      IF pcjfn THEN                                                   01330
         RETURN( MAX( jfnfdb[$fdbwrt].LH, pcfdb[$fdbwrt].LH) )        01331
      ELSE RETURN( jfnfdb[$fdbwrt].LH );                              01332
   RETURN( 35M );             % no grouping (or bad record) %       01333
                                                                      01334
   END.                                                               01335
```

% %

01893

```
(dlgets)     % return the sort key for this file %                    02119
   PROCEDURE                                                          02120
      (sortk,         % dlsort record %                               02121
      jfnfdb,         % address of the file fdb %                     02122
      pcjfn, % jfn for the pc if it exists %                         02123
      pcfdb  % address of the pc fdb %                               02124
      );                                                             02125
   REF jfnfdb, pcfdb;                                                02126
                                                                      02127
   % find out if any sorting %                                       02128
      IF NOT sortk THEN RETURN( 35M );        % no sorting %        02129
   IF sortk.dlsacc THEN       % sort by accounts %                   02130
      IF jfnfdb[$fdbact] < 0 THEN                                    02131
         RETURN( jfnfdb[$fdbact].RH )                                02132
      ELSE RETURN( jfnfdb[$fdbact] );                                02133
   IF sortk.dlsart THEN       % sort by archive tapes %              02134
      RETURN( jfnfdb[$fdbart].flsnat );                              02135
   IF sortk.dlstar THEN       % sort by archive time and date %      02136
      RETURN( jfnfdb[$fdbtsa] );                                     02137
   IF sortk.dlsbyt THEN       % sort by bytesize %                   02138
      RETURN( jfnfdb[$fdbbyv].flbyts );                              02139
   IF sortk.dlstcr THEN       % sort by time and date of creation %
                                                                      02140
      RETURN( jfnfdb[$fdbcrv] );                                     02141
   IF sortk.dlsdlt THEN       % sort by deletion status %            02142
      RETURN( jfnfdb[$fdbctl].fdbdel );                              02143
   IF sortk.dlstdm THEN       % sort by time and date of dump %      02144
      RETURN( jfnfdb[$fdbtdm] );                                     02145
   IF sortk.dlsdmt THEN       % sort by dump tape %                  02146
      RETURN( jfnfdb[$fdbbkf].fldmpt );                              02147
   IF sortk.dlsdfr THEN       % sort by file retention specs %       02148
      RETURN( jfnfdb[$fdbbyv].fldfr );                               02149
   IF sortk.dlslwr THEN       % sort by last writer %                02150
      IF pcjfn THEN RETURN( pcfdb[$fdbuse].LH )                      02151
      ELSE RETURN( jfnfdb[$fdbuse].LH );                             02152
   IF sortk.dlslen THEN       % sort by length in bytes %           02153
      IF pcjfn THEN RETURN( jfnfdb[$fdbsiz] + pcfdb[$fdbsiz] )       02154
      ELSE RETURN( jfnfdb[$fdbsiz] );                                02155
   IF sortk.dlsnac THEN       % sort by number of accesses %        02156
      IF pcjfn THEN                                                  02157
         RETURN(                                                     02158
            jfnfdb[$fdbcnt].flnmrd + jfnfdb[$fdbcnt].flnmwr +        02159
            pcfdb[$fdbcnt].flnmrd + pcfdb[$fdbcnt].flnmwr )          02160
      ELSE RETURN( jfnfdb[$fdbcnt].flnmrd + jfnfdb[$fdbcnt].flnmwr );
                                                                      02161
   IF sortk.dlsnrd THEN       % sort by number of reads %           02162
      IF pcjfn THEN                                                  02163
         RETURN( jfnfdb[$fdbcnt].flnmrd + pcfdb[$fdbcnt].flnmrd )
                                                                      02164
      ELSE RETURN( jfnfdb[$fdbcnt].flnmrd );                         02165
   IF sortk.dlsnwr THEN       % sort by number of writes %          02166
      IF pcjfn THEN                                                  02167
         RETURN( jfnfdb[$fdbcnt].flnmwr + pcfdb[$fdbcnt].flnmwr )
                                                                      02168
      ELSE RETURN( jfnfdb[$fdbcnt].flnmwr );                         02169
   IF sortk.dlstov THEN       % sort by orig. ver. creation t & d %
```

```
                                                                    02170
   RETURN( jfnfdb[$fdbcre] );                                       02171
IF sortk.dlstrd THEN      % sort by last read time and date %
                                                                    02172
   IF pcjfn THEN RETURN( MAX( jfnfdb[$fdbref], pcfdb[$fdbref]) )
                                                                    02173
   ELSE RETURN( jfnfdb[$fdbref] );                                  02174
IF sortk.dlssiz THEN      % sort by size in pages %                  02175
   IF pcjfn THEN                                                    02176
      RETURN( jfnfdb[$fdbbyv].flpgsz + pcfdb[$fdbbyv].flpgsz )
                                                                    02177
   ELSE RETURN( jfnfdb[$fdbbyv].flpgsz );                          02178
IF sortk.dlstwr THEN      % sort by last write time and date %
                                                                    02179
   IF pcjfn THEN RETURN( MAX( jfnfdb[$fdbwrt], pcfdb[$fdbwrt]) )
                                                                    02180
   ELSE RETURN( jfnfdb[$fdbwrt] );                                  02181
RETURN( 35M );    % bad input, thus no sort %                      02182
                                                                    02183
END.
                                                                    02184
%  %                                                               02185
```

```
(dlinfb)     % insert a dlfb in chain %                                  02186
   PROCEDURE                                                            02187
      (chns, % address of chain start %                                 02188
      adlmb, % address of directory list master block %                 02189
      adlfb, % address of dlfb %                                        02190
      sortk  % sort specification %                                     02191
      );                                                                02192
   LOCAL                                                                02193
      gptr,  % current group block pointer %                            02194
      lptr,  % current position pointer %                               02195
      tptr   % temporary pointer %                                      02196
      ;                                                                 02197
   REF chns, adlmb, adlfb, lptr, tptr, gptr;                            02198
                                                                        02199
   % find the right directory group block %                             02200
      IF NOT chns THEN       % no group chain exists yet %             02201
         BEGIN                                                          02202
         % create group block for the group %                          02203
         % if no room return since we can't do anything %              02204
            IF NOT ([&chns] _ dlgtblk(&adlmb, dlgbl)) THEN              02205
               RETURN( FALSE );                                         02206
         % make this the group block for this group %                  02207
            [chns].dlgbgk _ adlfb.dlfbgk;                               02208
         % set up current pointer %                                    02209
            &lptr _ [&chns];                                            02210
         END                                                            02211
      ELSE                                                              02212
         CASE adlfb.dlfbgk OF                                           02213
            < [chns].dlgbgk: % this group fits in front of first
            group %                                                     02214
               BEGIN                                                    02215
               % get new group block (if can't then return) %           02216
                  IF NOT (&tptr _ dlgtblk(&adlmb, dlgbl)) THEN          02217
                     RETURN( FALSE );                                   02218
               % set up this block to be for this group %              02219
                  tptr.dlgbgk _ adlfb.dlfbgk;                           02220
               % chain in new block & fix up start of chain pointer %
                                                                        02221
                  tptr.dlgbnb _ [&chns];                                02222
                  [chns].dlgbpb _ &tptr;                                02223
                  [&chns] _ &lptr _ &tptr;                              02224
               END;                                                     02225
            = [chns].dlgbgk: % this file is in first group %            02226
               &lptr _ [&chns];                                         02227
            > [chns].dlgbgk: % this group fits after first group %
                                                                        02228
               BEGIN                                                    02229
               % set up current pointer to loop through chain %        02230
                  &lptr _ [&chns];                                      02231
               LOOP                                                     02232
                  IF NOT lptr.dlgbnb THEN                               02233
                     % chain end, add new group (return if no space)
                     %                                                  02234
                     BEGIN                                              02235
                     IF NOT (&tptr _ dlgtblk(&adlmb, dlgbl)) THEN
                                                                        02236
```

```
                        RETURN( FALSE );                                02237
                     tptr.dlgbgk _ adlfb.dlfbgk;                        02238
                     tptr.dlgbpb _ &lptr;                               02239
                     lptr.dlgbnb _ &tptr;                               02240
                     &lptr _ &tptr;                                     02241
                     EXIT LOOP;                                         02242
                     END                                                02243
                  ELSE                                                  02244
                     % not chain end, look at next block %              02245
                     BEGIN                                              02246
                     &lptr _ lptr.dlgbnb;                               02247
                     CASE adlfb.dlfbgk OF                               02248
                        < lptr.dlgbgk: % before this one (after last)
                        %                                               02249
                           BEGIN                                        02250
                           IF NOT (&tptr _ dlgtblk(&adlmb, dlgbl))
                           THEN                                         02251
                              RETURN( FALSE );                          02252
                           tptr.dlgbgk _ adlfb.dlfbgk;                  02253
                           tptr.dlgbnb _ &lptr;                         02254
                           tptr.dlgbpb _ lptr.dlgbpb;                   02255
                           lptr.dlgbpb _ [lptr.dlgbpb].dlgbnb _
                           &tptr;                                       02256
                           &lptr _ &tptr;                               02257
                           EXIT LOOP;                                   02258
                           END;                                         02259
                        = lptr.dlgbgk: % this is the one! %             02260
                           EXIT LOOP;                                   02261
                        > lptr.dlgbgk: % greater, continue search %
                                                                        02262
                           REPEAT LOOP;                                 02263
                        ENDCASE;     % can't possibly happen %          02264
                     END;                                               02265
               END;                                                     02266
            ENDCASE; % can't possibly happen %                          02267
% by now lptr points to proper group block %                          02268
% find proper sort place within this group %                          02269
   IF NOT lptr.dlgbsc THEN                                              02270
      BEGIN                % no entries yet this group %                02271
      % make this the first one (and we're done) %                     02272
         lptr.dlgbsc _ lptr.dlgbnu _ &adlfb;                            02273
      RETURN( TRUE );                                                   02274
      END                                                               02275
   ELSE                                                                 02276
      CASE adlfb.dlfbsk OF                                              02277
         >= [lptr.dlgbnu].dlfbsk: % goes after last entry %             02278
            BEGIN                                                       02279
            adlfb.dlfbpb _ lptr.dlgbnu;                                 02280
            lptr.dlgbnu _ [lptr.dlgbnu].dlfbnb _ &adlfb;                02281
            RETURN( TRUE );                                             02282
            END;                                                        02283
         ENDCASE                    % goes before last entry %          02284
            BEGIN                                                       02285
            % save pointer to this group block %                        02286
               &gptr _ &lptr;                                           02287
            % move current pointer to last entry %                      02288
```

```
            &lptr _ lptr.dlgbnu;                                    02289
         LOOP                                                       02290
            IF NOT lptr.dlfbpb THEN    % hit end of chain %        02291
               BEGIN                                                02292
               adlfb.dlfbnb _ &lptr;                                02293
               lptr.dlfbpb _ &adlfb;                                02294
               gptr.dlgbsc _ &adlfb;                                02295
               RETURN( TRUE );                                      02296
               END                                                  02297
            ELSE                                  % not chain end
            yet %                                                   02298
               BEGIN                                                02299
               % step over to next entry and see what happens %
                                                                    02300
                  &lptr _ lptr.dlfbpb;                              02301
                  CASE adlfb.dlfbsk OF                              02302
                     >= lptr.dlfbsk:   % after this one %           02303
                        BEGIN                                       02304
                        adlfb.dlfbpb _ &lptr;                       02305
                        adlfb.dlfbnb _ lptr.dlfbnb;                 02306
                        lptr.dlfbnb _                               02307
                           [lptr.dlfbnb].dlfbpb _ &adlfb;           02308
                        RETURN( TRUE );                             02309
                        END;                                        02310
                     ENDCASE REPEAT LOOP; % before this one,
                     loop %                                         02311
               END;                                                 02312
         END;                                                       02313
                                                                    02314
END.
                                                                    02315
% %                                                                 02316
```

```
(dlsnps)     % directory command 2 pass - get rid of redundant PCs %
                                                                        01550
   PROCEDURE                                                            01551
      (nlschn,       % address 1st dlfb of nls file with a pc %         01552
      pcchn  % address 1st dlfb of pc file %                            01553
      );                                                                01554
   LOCAL                                                                01555
      tpcchn;        % loop variable %                                  01556
   REF nlschn, pcchn, tpcchn;                                           01557
                                                                        01558
   % special quick exit %                                              01559
      IF NOT &pcchn THEN RETURN;                                        01560
   % loop thru nls chain %                                             01561
      WHILE &nlschn # 0 DO                                              01562
         BEGIN                                                          01563
         &tpcchn _ &pcchn;                                              01564
         WHILE &tpcchn # 0 DO                                           01565
            BEGIN                                                       01566
            IF NOT tpcchn.dlfbfl.dlstig THEN                            01567
               IF *[nlschn.dlfbap]* = *[tpcchn.dlfbap]* THEN            01568
                  BEGIN                                                 01569
                  tpcchn.dlfbfl.dlstig _ TRUE;                          01570
                  EXIT LOOP;                                            01571
                  END;                                                  01572
            &tpcchn _ tpcchn.dlfbcp;                                    01573
            END;                                                        01574
         &nlschn _ nlschn.dlfbcp;                                       01575
         END;                                                           01576
   RETURN;                                                              01577
                                                                        01578
   END.
                                                                        01579
   %  %                                                                 01897
```

```
(dickdl)     % check deletion status of this file %                    01580
   PROCEDURE                                                          01581
      (fjfn, % jfn of the file %                                      01582
      delbit % request bits %                                         01583
      );                                                              01584
                                                                      01585
   CASE delbit OF                                                     01586
      = 1:   % deleted only %                                         01587
         BEGIN                                                        01588
         !gtfdb( fjfn, 186+1, $r3);                                   01589
         IF r3.fdbdel THEN RETURN( TRUE )                             01590
         ELSE RETURN( FALSE );                                        01591
         END;                                                         01592
      ENDCASE RETURN( TRUE );  % wouldn't have jfn if false %         01593
                                                                      01594
   END.
                                                                      01595
   %  %                                                              01896
```

```
(dlblds)     % build string for second pass ( and set bits in dlfb %
                                                                    01596
   PROCEDURE                                                        01597
      (fjfn, % jfn of the file %                                    01598
      pcjfn, % jfn of the pc if it exists %                         01599
      string,        % address of string to receive the information %
                                                                    01600
      adlfb  % address of the dlfb %                                01601
      );                                                            01602
   LOCAL STRING                                                     01603
      locstr[40];    % temp string %                                01604
   REF string, adlfb;                                               01605
                                                                    01606
   % if no pc then see if this is a pc %                           01607
      IF NOT pcjfn THEN                                             01608
         BEGIN                  % find out if this is a pc %       01609
         % get extension name %                                     01610
            jfntostr( fjfn, $locstr, 000100B6);                     01611
         % now see if it is PC %                                    01612
            IF *locstr* = "PC" THEN                                 01613
               BEGIN % this is a pc %                               01614
               adlfb.dlfbfl.dlstpc _ TRUE;                          01615
               RETURN;                                              01616
               END                                                  01617
            ELSE RETURN;      % not a PC %                         01618
                                                                    01619
         END;                                                       01620
   % has a pc so setup and return %                                01621
      adlfb.dlfbfl.dlstnl _ TRUE;                                   01622
      jfnflink( pcjfn, $string, 011110B6+1);                        01623
      RETURN;                                                       01624
                                                                    01625
   END.
                                                                    01626
   %  %                                                           01898
```

```
(dlbgac)     % get account string or number for a directory list %
                                                                    01627
   PROCEDURE                                                        01628
      (fjfn, % jfn of the file %                                    01629
      fdb,   % address of file's fdb %                              01630
      adlmb  % address of dlmb %                                    01631
      );                                                            01632
   LOCAL                                                            01633
      index; % temp for concatenating strings %                     01634
   LOCAL STRING                                                     01635
      tstring[40];   % temp to hold alphanumeric account strings %
                                                                    01636
   REF fdb, adlmb;                                                  01637
                                                                    01638
   tstring.L _ tstring.M;                                           01639
   !gactf( fjfn, chbmty+$tstring);                                  01640
      GOTO gerr;              % something really screwed up %     01641
      GOTO acisstr;  % account is string %                          01642
      GOTO acisnum;  % account is number %                          01643
   (gerr):                                                          01644
      fdb[$fdbact] _ -1;                                            01645
      RETURN;                                                       01646
   (acisstr):                                                       01647
      FOR index _ 1 UP UNTIL > tstring.L DO                         01648
         IF *tstring*[index] # 0 THEN NULL                          01649
         ELSE                                                       01650
            BEGIN                                                   01651
            tstring.L _ index _ index - 1;                          01652
            EXIT LOOP;                                              01653
            END;                                                    01654
      IF index <= 0 THEN                                            01655
         BEGIN                                                      01656
         fdb[$fdbact] _ -1;                                         01657
         RETURN;                                                    01658
         END;                                                       01659
      IF NOT (fdb[$fdbact] _ dlgtblk( &adlmb, (index+4)/5+1) ) THEN
                                                                    01660
         fdb[$fdbact] _ -1                                          01661
      ELSE                                                          01662
         BEGIN                                                      01663
         [fdb[$fdbact]].M _ index %-1%;  %-1 makes string too short%
                                                                    01664
         *[fdb[$fdbact]]* _ *tstring*;                              01665
         fdb[$fdbact] _ fdb[$fdbact] .V 5B11;                       01666
         END;                                                       01667
      RETURN;                                                       01668
   (acisnum):                                                       01669
      fdb[$fdbact] _ r2.RH;                                         01670
      RETURN;                                                       01671
                                                                    01672
   END.
                                                                    01673
   %  %                                                           01899
```

```
(dlbgtd)     % append [time and] date string to passed string % 01674
   % NOTE: "time" is internal date and time, is expected to be in
   TOPS20 format on TOPS20 machines.  This routine is specifically
   for times gotten from a file FDB.%                              03190
   PROCEDURE                                                       01675
      (param,        % field indicating date or time and date request
      %                                                            01676
      time,  % time and date %                                     01677
      string % string to be appended to %                          01678
      );                                                           01679
   LOCAL STRING                                                    01680
      locstr[40];    % temp string %                               01681
   REF string;                                                     01682
                                                                   01683
   % catch any errors %                                           01684
      ON SIGNAL                                                    01685
         ELSE                                                      01686
            BEGIN                                                  01687
            *string* _ *string*, "Can't Convert [Time and] Date";
                                                                   01688
            RETURN;                                                01689
            END;                                                   01690
   % now convert time appropriately %                             01691
      IF tops20flag THEN time _ totentad(time);  %convert to TENEX
      format%                                                      03191
      CASE param OF                                                01692
         =1: % date only %                                         01693
            datfrmt( time, 40186, $locstr);                        01694
         ENDCASE     % time and date %                             01695
            datfrmt( time, 186, $locstr);                          01696
   % append the string and return %                               01697
      *string* _ *string*, *locstr*;                               01698
      RETURN;                                                      01699
                                                                   01700
   END.
                                                                   01701
   %  %                                                          01901
```

```
(dlbgpr)     % get protection string for individuall field %     01702
   PROCEDURE                                                    01703
      (prot, % protection bits for this field %                 01704
      string % address of string to be appended to %            01705
      );                                                        01706
   REF string;                                                  01707
                                                                01708
   % print the numeric protection first %                       01709
      *string* _ *string*, '(, STRING( prot, 8), ") - ";        01710
   % now interpret the bits %                                   01711
      IF NOT prot THEN                                          01712
         *string* _ *string*, "No Access "                      01713
      ELSE                                                      01714
         BEGIN                                                  01715
         IF (prot.flprre := FALSE) THEN                         01716
            IF prot THEN *string* _ *string*, "Read, "          01717
            ELSE *string* _ *string*, "Read ";                  01718
         IF (prot.flprwr := FALSE) THEN                         01719
            IF prot THEN *string* _ *string*, "Write, "         01720
            ELSE *string* _ *string*, "Write ";                 01721
         IF (prot.flprex := FALSE) THEN                         01722
            IF prot THEN *string* _ *string*, "Execute, "       01723
            ELSE *string* _ *string*, "Execute ";               01724
         IF (prot.flprap := FALSE) THEN                         01725
            IF prot THEN *string* _ *string*, "Append, "        01726
            ELSE *string* _ *string*, "Append ";                01727
         IF prot.flprli THEN *string* _ *string*, "List ";      01728
         END;                                                   01729
   % finish up and return %                                     01730
      *string* _ *string*, "Allowed";                           01731
      RETURN;                                                   01732
                                                                01733
   END.                                                         01734
   %  %                                                        01900
```

```
(dlbglw)     % get last writer string %                                  01735
   PROCEDURE                                                            01736
      (dirno,         % directory number of last writer or zero %
                                                                        01737
      string,         % string to get info %                           01738
      jfn    %ver. 3 uses jfn instead (dirno would be zero)%           03200
      );                                                                01739
   LOCAL STRING                                                         01740
      dirname[40];    % temp string to get directory name %            01741
   REF string;                                                          01742
                                                                        01743
   % catch any errors %                                                01744
      ON SIGNAL                                                         01745
         ELSE                                                           01746
            BEGIN                                                       01747
            *string* _ *string*, "can't convert directory number";
                                                                        01748
            RETURN;                                                     01749
            END;                                                        01750
   % check for null field %                                            01751
      IF NOT dirno AND NOT jfn THEN                                     01752
         BEGIN                                                          03205
         *string* _ *string*, "SYSTEM";                                 01754
         RETURN;                                                        01755
         END;                                                           03206
   % get the string %                                                  03223
      IF NOT dirno AND t2v3flag THEN                                    03208
         BEGIN                                                          03209
         !gfust(1B6+jfn, chbmty+$dirname);                              03212
         END                                                            03219
      ELSE                                                              03220
         BEGIN                                                          03221
         gdname(dirno, $dirname);                                       01758
         END;                                                           03222
   % add it to passed string %                                         01757
      *string* _ *string*, *dirname*;                                   01759
   % done so return %                                                  01760
      RETURN;                                                           01761
                                                                        01762
   END.
                                                                        01763
   %  %                                                                01902
```

```
(dlbgas)     % get archive status string %                              01764
   PROCEDURE                                                           01765
      (flags,        % flags indicating archive status %               01766
      string % address of string to get archive status %               01767
      );                                                               01768
   REF string;                                                         01769
                                                                       01770
   IF NOT                                                              01771
      (flags.fdbaar .V flags.fdbadl .V flags.fdbnar .V flags.fdbarc)
      THEN BEGIN                                                       01772
      *string* _ *string*, "None";                                     01773
      RETURN;                                                          01774
      END;                                                             01775
   IF flags.fdbaar THEN                                                01776
      *string* _ *string*, "Already Archived  ";                       01777
   IF flags.fdbadl THEN                                                01778
      *string* _ *string*, "Don't Delete After Archiving  ";           01779
   IF flags.fdbnar THEN                                                01780
      *string* _ *string*, "Archive Not Allowed  ";                    01781
   IF flags.fdbarc THEN                                                01782
      *string* _ *string*, "Archive Pending";                          01783
   RETURN;                                                             01784
   END.
                                                                       01785
FINISH of execfl                                                         02
```

FCONST

```
< NLS, FCONST.NLS.17, >, 29-Mar-78 16:43 JDH ;;;;
FILE fconst    % L10  <REL-NLS>fconst %% (L10,) (rel-nls,fconst.rel,) %
                                                                      0225
DECLARE EXTERNAL STRING                                               0223
   intca= "Type CA to proceed";                                       0224
DECLARE EXTERNAL                                                       06
   cmdelete= 4934768,                                                  07
   popstate= 12345,                                                    08
   interperr=3614B,        % cfd's extension %                        09
   gaderr=   427,          % TNLS address error %                     010
   statesig= 7453342B,    % For simulated GOTO STATE's%               011
   errsig=   0 ;                                                      012
DECLARE EXTERNAL  % case shift 2 mouse-keyset translation table %     0226
   cssh2= (                                                           0227
      "!, "", "#, "$, "%,                                             0228
      "&, "', "(, "), "@,                                             0229
      "+, "-, "*, "/, "^,                                             0230
      "0, "1, "2, "3, "4,                                             0231
      "5, "6, "7, "8, "9,                                             0232
      "=                                                              0233
   );                                                                 0234
DECLARE EXTERNAL  % mouse button translation table %                  0235
   msetbl= (                                                          0236
      0,                                                              0237
      CA,                                                             0238
      CD,                                                             0239
      C.,                                                             0240
      BC,                                                             0241
      ALT,                                                            0242
      BW,                                                             0243
      0                                                               0244
   );                                                                 0245
SET EXTERNAL %character codes and group names%                         013
   %character codes for special characters%                            014
      ascbst=   13B, % backspace stmt %                                015
      asctsw=  36B,  % text switch (used by todas execute text) %      016
      ctla=     1,    % control a %                                    017
      ctld=     4,    % control d %                                   0334
      ctle=     5,    % control e %                                    018
      ctlf=     6,    % control f %                                    019
      ctlg=     7,    % control g %                                    020
      ctln=     16B,  % control n %                                    021
      ctlo=     17B,  % control o %                                    022
      ctlp=     20B,  % control p %                                    023
      ctlr=     22B,  % control r %                                    024
      ctls=     23B,  % control s %                                    025
      ctlt=     24B,  % control t %                                    026
      ctlu=     25B,  % control u %                                    027
      ctlv=     26B,  % control v %                                    028
      ctlw=     27B,  % control w %                                    029
      ctlx=     30B,  % control x %                                    030
      ctly=     31B,  % control y %                                    031
      ctlz=     32B,  % control z %                                    032
      ascalt=   33B;  % altmode %                                      033
% TENEX things %                                                      039
   DECLARE EXTERNAL                                                   040
```

```
   % file access bit definitions %                                    041
      racc=100000B6,     % read access(bit 2) %                      042
      wacc=40000B6,      % write access(bit 3) %                     043
      eacc=20000B6,      % execute access(bit 4) %                   044
      tuacc=1000B6,      % trap to user on access(bit 8) %           045
      cwacc=400B6;       % copy on write access(bit 9) %             046
% File Things %                                                      047
   DECLARE EXTERNAL STRING  %file name extensions%                   051
      nlsext= "NLS",                                                 052
      savext= "SAV",                                                 053
      exeext= "EXE",                                                0338
      ctlext= "CTL";                                                 054
   DECLARE EXTERNAL                                                  055
      %flag values for GTJFN JSYS%                                  0331
         gtjoof=4B11,    %output file (default version next higher)%  056
         gtjoif=1B11,    %old file only%                            0273
         gtjprf=1B11,    %old file only%                            0272
         gtjoosf=4B11,   %output file (default version next higher)% 0271
         gtjoisf=1B11,   %old file only%                            0269
         gtjstr= 1B8,    %accept file group descriptor, determining name
         according to old/new file bits%                            0270
         gtjprv= 2B9,    %no access by other forks%                 0268
         gtjidl= 1B9,    %ignore deleted bit%                       0267
         gtjnfo= 2B11,   %new file only%                            0266
         %following 4 flag values applied to LH of flag word by
         getgtjflg procedure%                                       0333
         gtjwrt = 4B5,   %output file (like 4B11)%                  0325
         gtjred = 1B5,   %old file only (like 1B11)%                0327
         gtjigdel = 1B3,         %ignore deleted bit (like 1B9)%    0328
         gtjcpc = 2B3,   %no access by other forks (like 2B9)%      0326
         sgtjff = 1B6,   %do short form of GTJFN%                   0329
         jfnstf = 11110040001B, %for JFNS (jfn to string) JSYS;
         specifies format%                                          0330
      %file access types%                                           0332
         write= 0,       %write access only%                         057
         read= 1,        %read access only%                         0265
         readwrite= 2,   %read and write access only%               0264
         append= 3,      %append access only%                       0263
         rthawed= 5,     %read thawed access only%                  0262
         rwthawed= 4;    %read and write thawed access only%        0261
% parser %                                                           058
   % EXTERNL KEYWORDS (used as types for the SELECTION processor) %  059
   % SUBSYSTEM PROCESSING MODES %                                    092
      DECLARE EXTERNAL                                              093
         sbstart= 1,     % subsystem initialization %                094
         sbrun= 2,       % command execution of subsystem %          095
         sbfinish= 3,    % termination of a subsystem %              096
         sbpop= 4,       % backup after termination %                097
         sbrentry= 5;    % rentry after leaving a subsystem %        098
   % COMMAND RECOGNITION MODES %                                     099
      DECLARE EXTERNAL                                              0100
         mexpert= vexpert, % default, NLS style %                   0101
         mfixed= 5,      % fixed number of chars %                  0102
            fixl= 3,     % number of chars required for fixd mode %  0103
         mdemand= 2,             % TELNET model with right delimiters %
                                                                    0104
```

```
      manticipitory= 3;       % minimum unique, no right delimiters %
% COMMAND HERALD MODES %
   DECLARE EXTERNAL
      onechar= 1,             % only single character hearald %
      multchar= vmultchar;    % multiple character hearald
         (length is hrldsize) %
% INTERPRETER PARSING MODES %
   DECLARE EXTERNAL
      parsing= 1,             % normal parsing mode %
      pnext= 2,               % get successor mode %
      cleanup= 3,             % termination of a command %
      rptparse= 4,    % repeat of a command %
      backup= 5,              % backup after parse fail %
      popselect= 6,   % pop one selection %
      parsehelp= 7,   % solicit help info %
      parseqmark= 8; % solicit questionmark string %
% PROMPT MODES (VALUES FOR GLOBAL INPROMPTS) %
   DECLARE EXTERNAL
      noprompts=1,            % prompts off %
      partprompts=2, % no optional prompting %
      fullprompts=vfullprompts;       % all prompts %
% FEEDBACK MODES %
   DECLARE EXTERNAL
      verbsmode=vverbsmode,           % give em everything %
      tersemode=1;            % no noise word feedback, dont fill out
      keywords %
% FUNCTION CODES FOR FBCTL %
   DECLARE EXTERNAL
      clearcfl= 1001,         % clear cfl %
      addchar= 1002, % add char to cfl %
      keyword= 1003, % add keyword to cfl %
      startrec= 1004,         % start keyword recognition %
      echostr= 1005, % echo a given string in the feedback area %
      rechostr= 1006,         % replace a given string in the feedback
      area %
      typelit= 1007, % type out supplied string %
      startparams= 1008,      % begin parameter collection %
      incues= 1009,   % provide inpt prompts %
      fbpop= 1010,            % cleanup after poppiing states %
      typecalit= 1011,        % wait for CA after printing literal %
      fbaddlit= 1012,         % add string to literal feedback area %
      addcalit= 1013,         % wait for CA %
      typenulllit= 1014,      % type null literal string %
      fbendlit= 1015,         % type lit and wait for input %
      norstcalit= 1016;       %wait for CA but don't reset screen
      (another screenful coming)%
% RECORD SIZE DECLARATIONS %
   DECLARE EXTERNAL
      d2sel= 2,       % displacement of second selection %
      psellen= 4;     % length of pselrecord %
% user-options defaults %
   DECLARE EXTERNAL
```

```
   % view specs %                                                    0162
      defvs1,                 % default viewspecs - word 1 %        0163
      defvs2;                 % default viewspecs - word 2 %        0164
% Device things %                                                   0165
   DECLARE EXTERNAL                                                 0166
   % devices and device modes %                                     0167
      %device types%                                                0168
         ctty= 100B,          %controling tty number%               0169
         oprtty=1B,    %operator tty (gets journal errors, archive
         ret requests%                                              0335
            %changed by fintnls,saveit to arcoprtty, etc. for
            different hosts%                                        0336
         arcoprtty= -1,       %arc operator's tty number%          0170
            %-1 means don't send any messages to oprtty%            0337
         nsaoprtty=0,                                               0279
         dev33= 0,                                                  0172
         dev35= 1,                                                  0173
         dev37= 2,                                                  0174
         devtiex= 3,                                                0175
         devsri= 4,           %tasker displays%                     0176
         imlac0= 5,                                                 0177
         imlac1= 6,                                                 0178
         nettty=7,                                                  0179
         offline=101,                                               0180
         devlproc= 11,        %value of nldevice for line processors%
                                                                    0181
      %modes%                                                       0182
         typewriter=0,                                              0183
         fulldisplay=1;                                             0274
   %imlac protocal%                                                 0184
      DECLARE EXTERNAL                                              0185
         remfudge= 40B;                                             0275
      DECLARE EXTERNAL %control codes for remote displays (e.g.
      imlacs)%                                                      0186
         begmsg= 33B, %begin message character%                     0187
         echda= 3, %alternate seq da echoing%                       0188
         remsdda= 6, %suppress display area control character%      0189
         remrdda= 7, %restore display area control character%       0190
         remrsda= 11B, %restore string in display area control
         character%                                                 0191
         remtsn= 12B, %tty simulation on control character%         0192
         remtsf= 13B, %tty simulation off control character%        0193
         remlcm= 14B; %switch to long character mode%               0194
      DECLARE EXTERNAL                                              0195
         rinistr=(154663315466B, 154663315466B, 4048B9);            0276
            %string of 10 reminits plus 1 char%                     0196
   % Display stuff %                                                0197
      DECLARE EXTERNAL                                              0198
         syserr= 0,                                                 0199
         rtmax= 60;                                                 0277
      DECLARE EXTERNAL                                              0200
         current=-1;          %use current x,y position%           0201

                                                                    0202
   % jump return ring sizes%                                        0203
      DECLARE EXTERNAL                                              0204
```

```
        frrmax=25, %max entries in file return ring%                    0205
        srrmax=25; %max entries in statement return ring%               0206
    % Use measurement%                                                  0213
      DECLARE EXTERNAL                                                  0214
        dntyp=0, %indicates user running display NLS%                   0215
        tntyp=1, %indicates user running typewriter NLS%                0216
        nttyp=2,                                                        0217
        dspallf= 6;%reformat every stmt, all da's% %indicates user
        running network NLS%                                            0218
      DECLARE EXTERNAL                                                  0219
        orgstid= 2; %STID of origin with stfile=0%                      0220
%Measurement system constants%                                          0280
  DECLARE EXTERNAL                                                      0281
    mstu1=1,  %unconditional startup type%                              0282
    mstu2=2,  %unconditional startup type%                              0283
    mstg1=3, %first global sampling type%                               0284
    mstg2l=5, %last global sampling type%                               0322
    msstrlng=4,  %length of string to put out%                          0300
    msfmtmax=8;   %number of measurement types%                         0285
  DECLARE EXTERNAL  %measurement record types%                          0299
    mtruncerr = 0,                                                      0321
    mcomword = 1,                                                       0301
    mcombackup = 2,                                                     0302
    mdone = 3,                                                          0303
    mliteral = 4,                                                       0304
    maddrexp = 5,                                                       0305
    mbugmark = 6,                                                       0306
    mquestion = 7,                                                      0307
    minsert = 8,                                                        0308
    mrepeat = 9,                                                        0309
    mstop = 10,                                                         0310
    mbell = 11,                                                         0311
    msyntax = 12,                                                       0312
    mmousespecs = 13,                                                   0313
    mviewspecs = 14,                                                    0314
    mlevadj = 15,                                                       0315
    mloadavg = 16,                                                      0316
    mcomdel = 17,                                                       0317
    merrsig = 18,                                                       0318
    mpopstate = 19,                                                     0319
    mcompopstate = 20;                                                  0320
  DECLARE EXTERNAL    %flag file locations%                             0286
    ffacctime=500B,  %last flag file access time%                       0287
    ffprtab=510B,    %table of addresses of strings (or 0)%             0288
      %indexed by meas. type, msfmtmax long%                            0289
    fftftab=530B,   %table of overall type switches%                    0290
      %indexed by type, inhibits meas. if 0%                            0291
    ffrftab=540B,  %table of addresses of record flag tables%           0292
      %indexed by meas. type, 0 indicates use system defaults%          0293
    fffcnt=501B,      %meas. output file count for naming files%        0294
    ffglmod=502B,   %global sampling modulus%                           0295
      %also reserving 503B and 504B%                                    0324
    ffgicnt=505B,   %global sampling count%                             0296
      %also reserve 506B and 507B %                                     0323
    ffjobmax=577B,   %max jobno meas. system can handle%                0297
    ffjobltab=600B;  %measurement status table, indexed by jobno%       0298
```

```
% HELP constants %                                                    0248
   DECLARE EXTERNAL                                                  0250
      stkmax= 26,                                                    0278
      entind= -1,                                                    0251
      rngmax= 26,                                                    0252
      rnglnt= 2;                                                     0253
% systems cinits %                                                   0254
   DECLARE EXTERNAL                                                  0255
      nwheecinits=(1210400B, 320440B, 3061400B, 1016600B,            0256
         1313300B, 526640B, 1324640B,                                0257
         1207000B, 1323113B,                                         0258
         1430340B, 2326740B,                                         0339
         0,0,0,0,0,0,0,0,0,0);                                       0259
FINISH                                                               0221
   DECLARE EXTERNAL STRING % so only one copy exists in system These
   keywords are defined as external strings in CONST. %               060
      % STRUCTURAL ENTITIES %                                        061
         branch= "BRANCH",                                           062
         group= "GROUP",                                             063
         plex= "PLEX",                                               064
         statement= "STATEMENT",                                     065
      % TEXTUAL ENTITIES %                                           066
         character= "CHARACTER",                                     067
         invisible= "INVISIBLE",                                     068
         link= "LINK",                                               069
         number= "NUMBER",                                           070
         password= "PASSWORD",                                       071
         text= "TEXT",                                               072
         visible= "VISIBLE",                                         073
         word= "WORD",                                               074
      % MISC. ENTITIES %                                            075
         file= "FILE",                                               076
         oldfilelink= "OLDFILELINK",                                 077
         newfilelink= "NEWFILELINK",                                 078
         name= "NAME",                                               079
         return= "RETURN",                                           080
         filereturn= "FILERETURN",                                   081
         ident= "IDENT",                                            0247
         identlist= "IDENTLIST",                                    0246
         edge= "EDGE",                                               082
         marker= "MARKER",                                           083
         rest="REST", menu="MENU", hlpcom="HLPCOM", cntlq="CNTLQ",
         back="BACK"; % for HELP %                                  0249
```

FDATA

```
< NLS, FDATA.NLS.20, >, 19-Apr-78 16:20 JDH ;;;;
FILE fdata   %  L10   <REL-NLS>FDATA   %% (L10,) (rel-nls,FDATA.rel,) %
                                                                      0353
DECLARE EXTERNAL                                                      0348
   xacs[16],  % ACs at entry to NLS %                                 0349
   intmsi=0;  %flag for message at initialization time%               0350
DECLARE EXTERNAL STRING                                               0351
   intmsg[500]; %String for entry message%                            0352
REGISTER %here so DDT will use these on printout%                       03
   r1=1, r2=2, r3=3, r4=4, r5=5, p=7, wa=8, s=9, m=10, rp=11,           04
   a1=12, a2=13, a3=14, a4=15;                                          05
% stack sizes %                                                        06
   SET EXTERNAL                                                       08
      %call and multiple result stack lengths%                        09
         gstksz=  3072,  %length of general call stack%               010
         pstksz=  20,    %length of pattern stack%                    011
      %private stack for illegal intruction pseudo-interrupt%         012
         psisksz= 100;                                                013
% *** GLOBAL DATA FOR PAGE 0 *** %                                     014
% Special stuff declared in page 0, loaded at 140B %                   015
   %...KEEP THESE THINGS TOGETHER -- DONT ADD NEW DATA HERE...%        016
   DECLARE EXTERNAL %...important stuff...%                           017
   %...user related data...%                                          018
      cuno,    % current user number %                                020
      logdirno,   %login directory number%                          0526
         %version 3: will in general be different from cuno%        0527
      nldevice, %current terminal device%                             021
      console, % octal console number %                               022
      nwheelf=0, %psuedo-wheel flag%                                 0440
   %...set command and aptstr routine...%                             023
      xsmode=40B,    %  case set    %                                025
      modeon=0;      %  option mask %                                027
   % save stack for viewspec input backup%                            028
      DECLARE EXTERNAL STACK                                          029
         savevspec[20, 2];                                           0457
DECLARE EXTERNAL                                                      031
   nlcron=1; %measurement with nlscr jsys on/off%                     032
   % stack for illegal instruction pseudo-interrupt %                 034
      DECLARE EXTERNAL                                                035
         psistk[100];                                                0458
   %Bit tables for LSID and DAID assignment%                          036
      DECLARE EXTERNAL                                                037
         dabt[3],     %daid bit table%                                038
         lsbitables[56] %7 wrds, 8 das%,  %lsid bit tables for file text
         display areas, pointed to by dalsidb field in da records%    039
         clsbitables[56] %7 wrds, 8 das%,  %lsid bit tables for control
         info areas, pointed to by dalsidb field in da records%       040
         lsbtsize= 7,  %size (number of words) of a lsbitable%        041
         dabtsize= 3;  %size (number of words) of the dabt%           042
   % useroptions %                                                    043
      DECLARE EXTERNAL                                                044
         uojfn; % jfn for the user profile file %                     045
% DECLARE's %                                                         049
   DECLARE EXTERNAL                                                   050
      linkcnsl=0,     %line number of console when display linked%    052
      savnldevice,   %used to save nldevice during screen sharing%    053
```

```
   savedda,  %address of DNSL display area "Taken" to TNLS, CED%  054
   savetda,  %address of TNSL display area "Taken" to DNLS, CED%  055
   ttysim=0,   %daid of ttysimulation window in DNLS%             056
   ttyda,     %address od da currently being used as tty da%      057
   defttysim=1, %FLAG:                                            058
      TRUE: default tty-sim area being used,                      059
      FALSE: user-supplied tty-sim area being used%               060
   nlssbn,    %contains six-bit name for version of nls being used%
                                                                  063
   jnlsbn=0,  %contains six-bit name for journal%                 064
%...display area stuff.....%                                     065
   dpyarea[160], %dal*damax%                                      066
      dpyend,                                                     067
   nlsdas[160], %for cfl, name area, lit area, msg area, date-time
   area, view spec areas, etc%                                    068
      %CAN REDUCE THIS BY TWO*DAL WHEN MSGDA AND LITDA GO AWAY%   069
      nlsdae,                                                     070
   dacnt= 0,     %0=> no da's active%                             071
   rtfree= 0,    %free list pointer--physical entry count%        072
   vspsav[2],     %save previous viewspecs%                       073
   bmcnt=0,         %number of bugmarks on the screen%            074
   noclrall = 0,      %don't clear all windows; used by substitute to
   save the tty window%                                          0519
   bmsaved = 0,     %have any bugmarks been saved as stid and cc?%
                                                                  0497
%...pbug support.....%                                           075
   pbugtb[40],                                                    076
   pbugte,                                                        077
   pbugpt,                                                        078
%...system handles for display areas...%                         083
   tda,      %TNLS display area address%                         084
   tada,     %text area display area record address%             085
   cflda,    %Command Feedback Line display area record address% 086
   echoda,   %echo register (not used) display area record address%
                                                                  087
   ltvsda,   %L-T viewspec display area record address%           088
   vspcda,   %"hjuCP" view spec display area record address%     089
   namda,    %name area display area record address%             090
   subda,    %subsystem name display area record address%        091
   litda,    %literal feedback display area record address%      092
   msgda;    %message display area record address%               093
DECLARE EXTERNAL STRING                                          094
   rmdcr=     "^^^^^",     %initialize as up arrow%              095
   drmdcr=    "++++",      %initialize as plus%                  096
   spacestr=     "
   ",                                                             097
   altdir[30],    %alternate directories for rdlit%              0366
   altext[50],    %alternate extensions for rdlit%               0367
   cflstr[100],    %Command Feedback Line%                       098
   savcfl[100],     %previous Command Feedback Line%             099
   cflarw[25],      %arrow and question mark under CFL%          0100
   vspstr[30],   %viewspec string%                              0101
   initsr[15],   %user's ident%                                 0102
   nlsnam[10], %contains name of nls system being used%         0103
   cdstr1[2000],  %used by create display for intra-statement edits%
                                                                0104
```

```
   cdstr2[2000],  %used by create display for intra-statement edits%
                                                                    0105
   conreg[100],  %jump string save area%                            0106
   wrdreg[40],  %jump string save area%                             0107
   namreg[40],  %name default register%                             0108
   littabs[100]; %used in DNLS literal collection for tabs%         0109
DECLARE EXTERNAL                                                    0110
   sysbreak[128],    % break character transfer vector for rdlit %
                                                                    0111
   trnslo[128],      % typewriter output translation table %       0112
   trnsli[128];      % typewriter input translation table %        0113
DECLARE EXTERNAL                                                    0114
   altdfl, %flag indicating rdlit should use alternate directories %
                                                                    0368
   litdahandle,                                                     0115
   litline,    %index into column%                                  0116
   lplitline,    %saved value of litline for line processor
   terminals%                                                       0421
   lplitreset=0,    %reset literal flag for line processor%         0420
   litrmarg,   %column number of lit area rightmargin%              0117
   spacecol,   %column which had last SP (or non-printing char) for
   DNLS literal collection%                                         0118
   litstore,   %byte pointer for rapid string appends and line
   breaks%                                                          0119
   litstring,  %address of string being displayed in literal area%
                                                                    0120
   litcolumn[35],  %column array for DNLS literal collection%       0121
   tabstore;  %byte pointer into string TABS%                       0122
DECLARE EXTERNAL %...buff...%                                       0123
   buff[40],   %character input buffer -- array for speed%          0124
   buffs=  0,        %strating buffer index%                        0125
   buffn= 0, %next buffer index%                                    0126
   buffct,     %index into buff and bufffg%                         0127
   buffsz= 39; %max index into buff%                                0128
DECLARE EXTERNAL STACK                                              0131
   bugmarks[40],  %for bugmarks%                                    0132
   bmauxstk[40],  %for saving bugmarks as stid and cc during viewspec
   F and f%                                                         0496
   fvstk[65 %svmxlev+1%];                                           0459
%..... HELP Declarations..... %                                     0370
   DECLARE EXTERNAL                                                 0371
      helploc[4], % fake subsystem dispatch record %                0464
      calcaux,                                                      0463
      athelp=0,   % TRUE if at help command, FALSE otherwise. %
      inhelp=0;   % TRUE if in help command, FALSE otherwise. %    0462
   DECLARE EXTERNAL                                                 0375
      qagain=0,   % TRUE if in cntl-q typed while in help command,
      FALSE otherwise. %                                            0461
      hlpfileno,                                                    0467
      qda,                                                          0466
      qsw,                                                          0473
      qstorblk,                                                     0472
      qnewstmt,                                                     0471
      qdavspc,                                                      0465
      qdavs2,                                                       0470
      overscreen, % TRUE if screen overflows. %                     0380
```

```
    hseger= 0,  % TRUE if trouble in help sequence generator %
    multiflg=0,
    sflg=1      % search type %;
DECLARE EXTERNAL
    % Format characters %
      menchr= '],
      comchr= '%,
    % Format type of statement %
      mentyp=1,
      multyp=2,
      untyp=0;
DECLARE EXTERNAL
    moremenu,  % TRUE if more menu to be shown. %
    conrng,    % The address of the context ring block. %
    curstk,
    curindex,  % The address of the current context ring element %
    confre;    % The next free context index. %
DECLARE EXTERNAL
    newstk= 1; % flag for qmenu.  calls pushent for new context
    stack only if TRUE %
DECLARE EXTERNAL % Contexts for entry and top. %
    topcon[2],
    entcon[2];
DECLARE EXTERNAL
    qspecs,    % current query viewspecs in force %
    qpspecs;   % previous query viewspecs in force %
DECLARE EXTERNAL
    qmenumax= 15,
    qmenucnt= 0,  % current value for NEXT menu item %
    qcolwidth= 24;
DECLARE EXTERNAL
    hdebug= 1; % true for debugging...loads xhelp %
DECLARE EXTERNAL
    fwrd,      % true if the first word of a statement bugged %
    glblstd;   % the stid of the last branch search %
DECLARE EXTERNAL STRING
    failnm[100],  % name searched for but failed %
    defvspc[100]; % default viewspecs for a file %
%...frozen statement list...%
DECLARE EXTERNAL
fzlsts[80],
    fzlste, %end of buffer%
fzfree, %free list pointer for fzlsts%
frzmax= 20; %length of freeze list%
%...input stuff...%
DECLARE EXTERNAL STRING
    altinp[100]; %LP startup alternate input%
DECLARE EXTERNAL
    msdbit= 0,     % mouse button dirty bit %
    curmbt= 0,     % current state of mouse buttons %
    ordmbt= 0,     % or'd state of mouse buttons %
    gcords,              % coordinates %
    bugreg[2],     %bug mark area%
    cflpos,        %command feedback column counter%
```

```
      csrarmd=1,          %cursor armed-disarmed flag%                0160
      arowon=0,           %CFL arow on-off flag%                      0161
      ltsmall=1,          %lt-viewspec area large-small flag%         0162
      qmrkon=0,           %CFL question mark on-off flag%             0163
      namenull=1,         %TRUE if name area is null (empty)%         0164
      litreset=1,         %TRUE if literal area is reset%             0165
      littyflag=0,        %TRUE if lit area is being used as a tty
      window%                                                         0166
      litapflag=0,        %TRUE if lit area in append mode%           0167
      msgreset=1,         %TRUE if message area is reset%             0168
      namereset=1;    %TRUE if name area is reset%                    0169
DECLARE EXTERNAL                                                      0483
%...control environment stuff...%                                     0177
   inpwatchjfn= 0, %jfn for output control file%                      0178
   inpjfn= 0, %jfn for control file input; 0 means use controlling
   tty%                                                               0179
%...measurement...%                                                   0498
   msrrip, %global needed to hold record field%                       0502
   msjfn=0,  %jfn of measurement output file--0 indicates no
   measurement in progress %                                          0499
   msrtab=  %meas. record flag table%                                 0505
      (636736736440B, 636636676676B, 636636636636B,                   0506
      676676636636B, 636636636736B, 000000000736B),                   0507
         %these values are the defaults, overridden by flagfile
         optional entries%                                            0508
   mlastcpu = 0,  %last cpu time monitored%                           0509
   mlasttraps = 0,  %last number of page traps monitored%             0510
   mlastfaults = 0,  %last number of page faults monitored%           0511
   ldavgtime = 0,  %last time load average monitored%                 0512
   ldavgsec = 300,  %number of seconds between load average
   monitoring%                                                        0513
   mnaproc,  %table number of the no. of active processes table%      0514
   mdshare,  %table number of the job share table%                    0515
   msumrap,  %sum of reciprocals of no of. active processes%          0517
   msumrcnt,  %count of the number of units summed in msumrap%        0518
%...work station char input buffer...%                                0180
   curchr,     %curent char--after  markers have been resolved%       0181
   rawchr,     %address of lowest level character input routine%      0182
%...auxilary input stuff...%                                          0183
   auxsav,     % saved "rawchr" dispatch %                             0184
   auxmod,     % saved recogmode when reading aux. input %             0185
   auxmd2,     % saved recog2mode when reading aux. input %            0186
   auxinput=0, % flag to indicate the process is (not) in auxiliary
   input mode %                                                       0456
   auxtp1[2], % text ptr to start of auxilary text %                  0187
   auxtp2[2], % text ptr to end of auxilary text %                    0188
   auxwrk[7], %executable text work area%                             0189
%...global display/print parameters...%                               0190
   %...view parameters and values...%                                 0191
      %...View specs...%                                             0192
         sysvspec[2],    %view spec flags and values%                 0193
      lvdjnm,   %if true then print statement numbers during levadj %
                                                                      0194
   %...parameters to dpset for selective recreate display%            0199
      cdstd1,  %stid of stmt to be reformatted, if cdtype= dsprfmt
      or dsprfst; else passed as file indicators so da's may be
```

```
    selectively updated%                                                0200
    cdstd2,  %second stid or file indicator%                            0201
    cdstop,  %after encountering this stid, daupdate knows no
    further structural or editing changes took place on the old
    lsrt entries.%                                                      0202
    cdtype;  %option to inform daupdate of general nature of
    operation performaed by command issued.  Options are defined in
    data.%                                                              0203
DECLARE EXTERNAL                                                        0204
%...general global variables...%                                        0205
    continue=0, %  continue flag--set by terminate  %                   0206
    nlmode,    %= typewriter or NOT typewriter%                         0207
    nlparse=1, %TRUE for non-query entry; FALSE for query%              0208
    exquery= 0,              % no ident or initial file needed %         0209
    typefg= 1,     %type flag for alter%                                0210
    cmarkflg= 0,  %show control marker flag%                            0211
    echofg,         %typewriter echo flag for output%                   0212
    ctrlchar,                % if TRUE echo control chars as <^char>
    %                                                                   0213
    oshift,         %typewriter output shift mode%                      0214
    cshift,         %character which causes lower to upper case
    shift in TNLS for the following character -- see <inpfbk,
    tinptc>%                                                            0215
    wshift,         %character which causes lower to upper case
    shift in TNLS for the following word -- see <inpfbk, tinptc>%
                                                                        0216
    wordshift,      %Flag used by <inpfbk, tinptc> for doing word
    shifts%                                                             0217
    todlit,         %typewriter literal escape character%               0218
%...initial values for display areas...%                                0219
    rmarg, %right margin%                                               0220
    lmarg, %left margin%                                                0221
    tmarg, %top margin%                                                 0222
    bmarg, %bottom margin%                                              0223
    %text area%                                                         0224
       tabase= taleft,                                                  0225
       taleft= 0, %lmarg%                                               0226
       taright= 0, %rmarg%                                              0227
       tatop= 0, %tmarg + 9*tavinc%                                     0228
       tabottom= 0, %bmarg%                                             0229
       tacsize= 1, %character size%                                     0230
       % tafont= 0, font is normal%                                     0231
       tahinc= 0, %14*256%                                              0232
       tavinc= 0, %30*256%                                              0233
       tax= 0,                                                          0234
       tay= 0,                                                          0235
       tamind= 15,                                                      0369
    %Command Feedback Line (x=300B, y=1672B)%                           0236
       cflbase= cflleft,                                                0237
       cflleft= 0, %lmarg + 20*cflhinc%                                 0238
       cflright= 0, %cflleft + 40*cflhinc%                              0239
       cfltop= 0, %tmarg + 3*cflvinc%                                   0240
       cflbottom= 0, %cfltop + 2*cflvinc%                               0241
       cflcsize= 1, %character size%                                    0242
       cflfont= 0, %font is normal%                                     0243
       cflhinc= 0, %14*256%                                             0244
```

```
      cflvinc= 0, %30*256%                                              0245
      cflx= 0,                                                          0246
      cfly= 0,                                                          0247
   %Name area (x=600, y=1672B)%                                         0248
      nambase= namleft,                                                 0249
      namleft= 0, %cflright + cflhinc%                                  0250
      namright= 0, %rmarg%                                              0251
      namtop= 0, %cfltop%                                               0252
      nambottom= 0, %namtop + namvinc%                                  0253
      namcsize= 1, %character size%                                     0254
      namfont= 0, %font is normal%                                      0255
      namhinc= 0, %14*256%                                              0256
      namvinc= 0, %30*256%                                              0257
      namx= 0,                                                          0258
      namy= 0,                                                          0259
   %subsystem display%                                                  0260
      subbase= subleft,                                                 0261
      subleft= 0, %rmarg-16*subhinc%                                    0262
      subright= 0, %rmarg%                                              0263
      subtop= 0, %tmarg +  subvinc%                                     0264
      subbottom= 0, %subtop + subvinc%                                  0265
      subcsize= 1, %character size%                                     0266
      subfont= 0, %font is normal%                                      0267
      subhinc= 0, %14*256%                                              0268
      subvinc= 0, %30*256%                                              0269
      subx= 0,                                                          0270
      suby= 0,                                                          0271
   %L-T view spec area (x=0, y=1700B)%                                  0272
      ltbase= ltleft,                                                   0273
      ltleft= 0, %lmarg%                                                0274
      ltright= 0, %ltleft+15*lthinc%                                    0275
      lttop= 0, %tmarg + ltvinc%                                        0276
      ltbottom= 0, %lttop + ltvinc%                                     0277
      ltcsize= 0, %character size%                                      0278
      ltfont= 0, %font is normal%                                       0279
      lthinc= 0, %12*256%                                               0280
      ltvinc= 0, %30*256%                                               0281
      ltx= 0,                                                           0282
      lty= 0,                                                           0283
   %view spec area (x=0, y=1636)%                                       0284
      vsbase= vsleft,                                                   0285
      vsleft= 0, %ltleft%                                               0286
      vsright= 0, %ltright%                                             0287
      vstop= 0, %ltbottom + ltvinc%                                     0288
      vsbottom= 0, %vstop + vsvinc%                                     0289
      vscsize= 0, %character size%                                      0290
      vsfont= 0, %font is normal%                                       0291
      vshinc= 0, %12*256%                                               0292
      vsvinc= 0, %30*256%                                               0293
      vsx= 0,                                                           0294
      vsy= 0,                                                           0295
   %Literal feedback area%                                              0296
      litbase= litleft,                                                 0297
      litleft= 0, %taleft%                                              0298
      litright= 0, %taright%                                            0299
      littop= 0, %tatop - 3*tavinc%                                     0300
```

```
            litbottom= 0, %tabottom%                                    0301
            litcsize= 1, %character size%                               0302
            litfont= 0, %font is normal%                                0303
            lithinc= 0, %14*256%                                        0304
            litvinc= 0, %30*256%                                        0305
            litx= 0,                                                    0306
            lity= 0,                                                    0307
          %Message area%                                                0308
            msgbase= msgleft,                                           0309
            msgleft= 0, %cflleft%                                       0310
            msgright= 0, %dtleft -  msghinc%                            0311
            msgtop= 0, %tmarg + tavinc%                                 0312
            msgbottom= 0, %msgtop + msgvinc%                            0313
            msgcsize= 1, %character size%                               0314
            msgfont= 0, %font is normal%                                0315
            msghinc= 0, %14*256%                                        0316
            msgvinc= 0, %30*256%                                        0317
            msgx= 0,                                                    0318
            msgy= 0;                                                    0319
DECLARE EXTERNAL STRING %viewspecs%                                     0320
   vsstr1[10],      %L and T viewspec string%                           0321
   vsstr2[10];      %hjuCP viewspec string%                             0322
DECLARE EXTERNAL %line processor specific%                              0323
   dspjfn,              %JFN for line processor display%                0324
   ldspjfn,             %dspjfn for linked display%                     0325
   lpbaudfactor=1,      %baud rate factor -- MCS-4 to terminal%         0326
   lpxmax=79,           %max x for line processor display%              0327
   physxmax=79,         %physical xmax, used for slideing windows%      0524
   lpymax=26,           %max y for line processor display%              0328
   %see (nls,fintnls,getdev) concerning setting the following (and
   preceding) flags%                                                    0523
   lptype,              %type of line processor display%                0329
   lpdlpad,             %delete line padding for line processor display%
                                                                        0330
   lpilpad,             %insert line padding for line processor display%
                                                                        0441
   lptab= 40B,  %value of char to be shown as TAB%                      0331
   lppadchr= 177B,      %value of char to be used for delays%           0332
   newcp=0,             %TRUE if CP should use new protocol%            0521
      %also being used to indicate new graphics protocol%               0525
   scwindow=0,          %TRUE if has scroll window capability%          0522
   putbackchar=0,                                                       0333
      %TRUE if line-processor needs to have character rewritten when a
      bug mark is removed%                                              0334
   tracking,            %TRUE if line-processor is tracking mouse%      0335
   tracksend,   % count of  LP printer PSI's while not tracking %       0336
   lppch,        % pointer to lppch routine for given terminal type %
                                                                        0337
   lppcol,       % current column position for printer %                0338
   lpplin,       % current line number for printer %                    0339
   lppform,      % simulate formfeed flag %                             0340
   lpptimout,    % timeout cell for printer %                           0341
   lpppad,       % pad count for LF or what have you %                  0342
   lppldev,      % LP copy printer device code %                        0520
   lppjfn=0;     % jfn to copy to Lp printer %                          0343
   DECLARE EXTERNAL  %used in recording line processor errors (by
```

```
   ,inpfbk,errfill)%                                                    0425
      iperrec= 0,       %location of last LP error record%              0426
      iprng0,           %ring of date-times of last 10 LP errors%       0427
      iprng1,                                                           0428
      iprng2,                                                           0429
      iprng3,                                                           0430
      iprng4,                                                           0431
      iprng5,                                                           0432
      iprng6,                                                           0433
      iprng7,                                                           0434
      iprng8,                                                           0435
      iprng9,                                                           0436
      iprngp;           %pointer to next location in LP error ring%     0437
   DECLARE EXTERNAL STRING padstr=
   "
   ";                          0346
% declarations for use with the syntax genrating commands %             0403
   % global indicating current state of keyword recognition %           0414
      DECLARE EXTERNAL                                                  0415
         kwrstate;                                                      0484
            % < 0= => have recognized the keyword associated with the
            last entry on the path stack %                              0416
            %= 0= => have just entered keyword recognition mode, but
            have not recognized anything yet %                          0417
            % > 0= => have partially recognized the keyword associated
            with the last entry on the path stack %                     0418
   % this is storage for building a CML rule dynamically and for
   remembering how the rule was built %                                 0404
      DECLARE EXTERNAL                                                  0405
         synsubs[400],                                                  0406
         synstore[2],                                                   0489
         acursub,                                                       0488
         asupsub,                                                       0487
         syncur;                                                        0486
   % stack for  post-processing of unexpanded rules %                   0407
      DECLARE EXTERNAL                                                  0408
         pstkp= 0,                                                      0490
         pstk[50];                                                      0491
   % which commands do we get %                                         0409
      DECLARE EXTERNAL                                                  0410
         cmdctl= 0;                                                     0492
            % 0 - all commands %                                       0411
            % 1 - all DNLS commands %                                  0412
            % 2 - all TNLS commands %                                  0413
% declarations used in INPFDBK %                                       0422
   % symtflg IF true the cntrlgetchar simulates recorded timing when
   playing back a record file set in xplayback%                         0423
      DECLARE EXTERNAL                                                  0424
         symtflg= 0;                                                    0493
   % recrunfil IF TRUE implies that record seesion is to record a runfil
   useable file, i.e. no time stamping but merely record input
   characters %                                                         0438
      DECLARE EXTERNAL                                                  0439
         recrunfil= 0;                                                  0494
% declarations for use with measurement subsystem %                    0442
   DECLARE EXTERNAL                                                     0443
```

```
    analyzing=0,        %TRUE when performing measurements%          0444
    analtime=0,         %TRUE when performing measurements & at base
    level%                                                           0445
    notetime; %adr procedure to record time for measurements%        0446
FINISH of fdata                                                      0347
```

FILMNP

```
< NLS, FILMNP.NLS.24, >, 23-Apr-78 13:20 JDH ;;;;
FILE filmnp  % L10 <REL-NLS>filmnp %% (L10,) (rel-nls,filmnp.rel,) %  02
%.....FILE MANIPULATION SUPPORT ROUTINES.....%                        03
%Declarations%
                                                                      05
   DECLARE                                                            06
      newer = 0, older = 1, chkpcf = TRUE,                            07
      ckp1 = 2, ckp2 = 3, pc = 1;                                     08
   DECLARE STRING %for 5- to 7-bit character translation%             09
      trn5t7 = " ABCDEFGHIJKLMNOPQRSTUVWXYZ23456";                   010
   REGISTER                                                          011
      r1=1, r2=2, r3=3, p=7, wa=8, s=9, m=10, rp=11, a1=12,          012
      a2=13, a3=14, a4=15;                                           013
% Gets and stores %                                                02810
   % Get --  Structural relations %                                 02800
      (getsuc) %The stid for the successor field is returned.  If there
      is no successor, the stid of the up is returned.%
                                                                     0843
         PROCEDURE (stid);                                           0844
         LOCAL rngloc;                                               0845
         lodent(stid, rngtyp : rngloc);                              0846
         IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                     0847
         stid.stpsid _ [rngloc].rsuc;                                0848
         RETURN (stid) END.
                                                                     0849
      (getsub) %The STID in the sub field is returned.%
                                                                     0850
         PROCEDURE (stid);                                           0851
         LOCAL rngloc;                                               0852
         lodent(stid, rngtyp : rngloc);                              0853
         IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                     0854
         stid.stpsid _ [rngloc].rsub;                                0855
         RETURN (stid) END.
                                                                     0856
      (getail)                                                      01383
         %Given an stid, this procedure returns the stid of the tail of
         the current plex%                                          01384
         %-------------%                                            01385
         PROCEDURE (stid);                                          01386
         IF stid.stpsid # origin AND NOT getorf( stid ) THEN        01387
            UNTIL getftl(stid) DO                                   01388
            stid _ getsuc(stid);                                    01389
         RETURN(stid);                                              01390
         END.
                                                                    01391
      (getup)                                                       01392
         %Given an stid, this routine returns the stid of the source of
         the original stid%                                         01393
         %-------------%                                            01394
         PROCEDURE(stid);                                           01395
         IF stid.stpsid # origin AND NOT getorf( stid ) THEN        01396
            stid _ getsuc(getail(stid));                            01397
         RETURN(stid);                                              01398
         END.
```

```
                                                                    01399
(gethed)                                                            01400
   %Given an stid, this routine returns the head of the current
   plex; if it is passed the origin statement, it returns the
   origin%                                                          01401
   %-------------%                                                  01402
   PROCEDURE(stid);                                                 01403
   IF stid.stpsid # origin AND NOT getorf( stid ) THEN              01404
      stid _ getsub(getup(stid));                                   01405
   RETURN(stid);                                                    01406
   END.
                                                                    01407
(getprd)                                                            01408
   %Given an stid, this routine returns the predecessor; if the
   psid heads a plex, the stid itself is returned%                  01409
   %-------------%                                                  01410
   PROCEDURE (stid);                                                01411
   LOCAL presid, %stid of predecessor%                              01412
      sucsid; %stid of successor of presid%                         01413
   IF getfhd(stid) THEN RETURN(stid);                               01414
   presid _ gethed(stid);                                           01415
   UNTIL (sucsid _ getsuc(presid)) = stid DO                        01416
      presid _ sucsid;                                              01417
   RETURN(presid);                                                  01418
   END.
                                                                    01419
(getend)                                                            01420
   %This procedure returns the end of the branch headed by the
   stid passed it.%                                                 01421
   %-------------%                                                  01422
   PROCEDURE(stid);                                                 01423
   UNTIL (stid := getsub(stid)) = stid                              01424
   DO stid _ getail(stid);                                          01425
   RETURN(stid);                                                    01426
   END.
                                                                    01427
(getvnd) %find end of branch%                                       01481
   % This procedure finds the "end" of the branch begun by         01482
   the stid passed it; it expects, as a second argument, a          01483
   maximum level to be used in determining the end of the           01484
   branch.  If all statements in the branch fall beneath the        01485
   minimal level, it returns the the stid passed it. %              01486
   %-------------------%                                           01487
   PROC(stid, level);                                               01488
   LOCAL curlev, tmstid;                                            01489
   curlev _ getlev(stid);                                           01490
   UNTIL curlev >= level DO                                         01491
      BEGIN                                                         01492
      IF (stid := getsub(stid)) = stid THEN RETURN(stid);           01493
      stid _ getail(stid);                                          01494
      BUMP curlev;                                                  01495
      END;                                                          01496
   RETURN(stid);                                                    01497
   END.                                                             01498
                                                                    01499
(getbck)                                                            01428
```

```
      %This procedure finds the back of the stid of the statemment
      passed it.  It does not observe viewspecs.%                    01429
      %-------------%                                                01430
      PROCEDURE(stid);                                               01431
      IF stid.stpsid = origin OR getorf(stid) THEN RETURN(stid);
                                                                     01432
      IF getfhd(stid) THEN RETURN(getup(stid));                      01433
      stid _ getprd(stid);                                           01434
      IF (stid := getsub(stid)) = stid THEN RETURN(stid);            01435
      RETURN(getend(getail(stid)));                                  01436
      END.
                                                                     01437
   (getnxt)                                                          01438
      %This procedure finds the sequentially "next" statement, i.e.
      the substatement, successor, or successor of up, etc, of the
      stid passed as argument.  Ignores all viewspecs.%              01439
      %-----------------%                                           01440
      PROCEDURE (stid);                                              01441
      IF stid = endfil THEN err($"end of file exceeded");            01442
      IF (stid := getsub(stid)) = stid THEN                          01443
         % no substructure %                                         01444
         BEGIN                                                       01445
         LOOP                                                        01446
            BEGIN                                                    01447
            IF stid.stpsid = origin OR getorf(stid) THEN RETURN
            (endfil);                                                01448
            IF getftl(stid) = 0 THEN EXIT; % not a tail %            01449
            stid _ getsuc(stid);                                     01450
            END;                                                     01451
         stid _ getsuc(stid);                                        01452
         END;                                                        01453
      IF stid.stpsid = origin OR getorf(stid) THEN RETURN (endfil);
                                                                     01454
      RETURN (stid);                                                 01455
      END.                                                           01456
                                                                     01457
                                                                     01458
% Get --  flag values and field contents in ring %                   02801
   (getftl) %The logical value of the tail flag is returned.%
                                                                      0857
      PROCEDURE (stid);                                               0858
      LOCAL rngloc;                                                   0859
      lodent(stid, rngtyp : rngloc);                                  0860
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                      0861
      RETURN ([rngloc].rtf) END.
                                                                      0862
   (getfhd) %The logical value of the head flag is returned.%
                                                                      0863
      PROCEDURE (stid);                                               0864
      LOCAL rngloc;                                                   0865
      lodent(stid, rngtyp : rngloc);                                  0866
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                      0867
      RETURN ([rngloc].rhf) END.
                                                                      0868
```

```
   (getnmf) %The logical value of the name flag is returned.%
                                                                    0869
      PROCEDURE (stid);                                             0870
      LOCAL rngloc;                                                 0871
      lodent(stid, rngtyp : rngloc);                                0872
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0873
      RETURN ([rngloc].rnamef) END.
                                                                    0874
   (getnam) %The hash word for the statement name is returned.%
                                                                    0884
      PROCEDURE (stid);                                             0885
      LOCAL rngloc;                                                 0886
      lodent( stid, rngtyp : rngloc);                               0887
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0888
      RETURN ([rngloc].rnameh) END.
                                                                    0889
   (getsid) %The statement identifier for the statement name is
   returned.%
                                                                    0890
      PROCEDURE (stid);                                             0891
      LOCAL rngloc;                                                 0892
      lodent( stid, rngtyp : rngloc);                               0893
      %this does NOT call err if sid=0 so that sid can be checked by
      other programs to test a ring element for validity%           0894
      RETURN ([rngloc].rsid) END.
                                                                    0895
% Get first property stdb %                                        02802
   (getsdb) %The STDB associated with  the specified ring element is
   returned.  NOT TO BE USED TO GET THE STDB TO WHICH AN INFERIOR
   TREE IS CONNECTED!  USE GETPHED INSTEAD.  If the node is the
   origin of an inferior tree, this procedure returns FALSE.%
                                                                    0875
      PROCEDURE (stid);                                             0876
      LOCAL stdb, rngloc;                                           0877
      lodent(stid, rngtyp : rngloc);                                0878
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0879
      IF [rngloc].rtorgin THEN RETURN(0);                          03668
      stdb _ 0;                                                     0880
      IF (stdb.stpsdb _ [rngloc].rsdb) = 0 THEN RETURN(0);          0881
      stdb.stfile _ stid.stfile;                                    0882
      RETURN (stdb) END.
                                                                    0883
% Get next property stdb %                                        02803
   (getpsdb) PROCEDURE %***% ( stdb );                              02642
      % Given an stdb, this procedure finds the spsdb field of the
      property (the next in the chain and converts it into an stdb;
      if it is zero it returns zero. %                              02643
      LOCAL nxtprop, dbloc;                                         02644
      lodent( stdb, sdbtyp : dbloc);                                02645
      nxtprop _ 0;                                                  02646
      IF (nxtprop.stpsdb _ [dbloc].spsdb) = 0 THEN RETURN(0);       02647
      nxtprop.stfile _ stdb.stfile;                                 02648
      RETURN (nxtprop);                                             02649
```

```
      END.
                                                                    02650
% Get next property stdb %                                          03649
   (getitree) PROCEDURE %***% ( stdb );                             03650
      % Given an stdb, this procedure finds the sitpsid field of the
      property (the inferior tree) and converts it into an stid; if
      it is zero it returns zero. %                                 03651
      LOCAL itstid, dbloc;                                          03652
      lodent( stdb, sdbtyp : dbloc);                                03653
      itstid _ 0;                                                   03654
      IF (itstid.stpsdb _ [dbloc].sitpsid) = 0 THEN RETURN(0);      03655
      itstid.stfile _ stdb.stfile;                                  03656
      RETURN (itstid);                                              03657
      END.
                                                                    03658
% Get -- field values from data blocks %
                                                                     0806
   (getorg) %***%                                                   02740
      %Given an stid, this routine returns the origin of the node%
                                                                    02741
      %-------------%                                              02742
      PROCEDURE(stid);                                              02743
      LOCAL rnl;                                                    02744
      REF rnl;                                                      02745
      LOOP                                                          02746
         BEGIN                                                      02747
         IF stid.stpsid = origin OR getorf( stid ) THEN             02748
            EXIT LOOP;                                              02749
         stid _ getup(stid);                                        02750
         END;                                                       02751
      RETURN(stid);                                                 02752
      END.
                                                                    02753
   (getorf) %***% %The logical value of the origin flag is returned.%
                                                                    02754
      PROCEDURE (stid);                                             02755
      LOCAL rngloc;                                                 02756
      lodent( stid, rngtyp : rngloc);                               02757
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    02758
      RETURN ([rngloc].rtorgin) END.
                                                                    02759
   (getnmdl) %***% PROCEDURE (stid);                                02773
      LOCAL
         dlleft,                                                    02774
         dlright,                                                   02775
         sdb;                                                       02776
      REF sdb;                                                      02777
      IF NOT lodprop( stid, txttyp: &sdb) THEN                      02778
         BEGIN                                                      02779
         % No text block associated: no name delimiters.  Error? %  02780
                                                                    02781
         dlleft _ dlright _ 0;                                      02782
         END                                                        02783
      ELSE                                                          02784
         BEGIN                                                      02785
```

```
      dlleft _ sdb.slnmdl;                                          02786
      dlright _ sdb.srnmdl;                                         02787
      END;                                                          02788
   RETURN(dlleft, dlright);                                         02789
   END.                                                             02790
(getphed) %***%                                                     02760
   %Given an stid  (hopefully in an inferior tree), this routine
   returns the stdb of the property to which is attached.  If not
   in an inferior tree, returns false%                              02761
   %-------------%                                                  02762
   PROCEDURE(stid);                                                 02763
   LOCAL stdb, rngloc;                                              03670
   IF stid.stpsid # origin THEN                                     02764
      BEGIN                                                         02765
      stid _ getorg(stid);                                          02766
      IF stid.stpsid = origin THEN RETURN(FALSE);                   02767
      % The psdb field of the origin of an inferior tree points to
      the  property header. %                                       02768
         lodent(stid, rngtyp : rngloc);                             03671
         IF [rngloc].rsid = 0 THEN err($"Bad statement
         identifier");                                              03672
         IF [rngloc].rtorgin = 0 THEN err($"Inferior tree origin
         error");                                                   03677
         stdb _ 0;                                                  03673
         IF (stdb.stpsdb _ [rngloc].rsdb) = 0 THEN RETURN(0);
                                                                    03674
         stdb.stfile _ stid.stfile;                                 03675
         RETURN(stdb);                                              02769
      END                                                           02770
   ELSE RETURN(FALSE);                                              02771
   END.
                                                                    02772
(getint) %***% %Called with STDB, returns initials of associated
statement (left justified in word zero filled).%
                                                                    02689
   PROCEDURE (stdb);                                                02690
   LOCAL STRING string[5];                                          02691
   LOCAL sdbint, sdbloc;                                            02692
   lodent( stdb, sdbtyp : sdbloc);                                  02693
   *string* _ NULL;                                                 02694
   string[1] _ 0;                                                   02695
   IF (sdbint _ [sdbloc].sinit) .A 4B6 THEN %old syle inits%        02696
      string[1].oldint _ sdbint                                     02697
   ELSE trnsint(sdbint, $string);                                   02698
      %sdb initials in five-bit characters%                         02699
   RETURN(string[1]);                                               02700
   END.
                                                                    02701
(trnsint) PROCEDURE(initials, string);                               0820
   %Given a word of 5-bit initials, this routine translates them
   and appends them to the string whose address is passed.%         0821
   REF string;                                                      0822
   LOCAL char, charno, count;                                       0823
   count _ 0;                                                       0824
   UNTIL (count _ count + 1) > 4 DO                                 0825
      BEGIN                                                         0826
```

```
        charno _ CASE count OF                                          0827
           =1: initials.cint1;                                          0828
           =2: initials.cint2;                                          0829
           =3: initials.cint3;                                          0830
           ENDCASE initials.cint4;                                      0831
        IF (char _ *trn5t7*[charno + 1]) = SP THEN EXIT LOOP            0832
        ELSE *string* _ *string*, char;                                 0833
        END;                                                            0834
     RETURN;                                                            0835
     END.

                                                                        0836
  (gettim) %***% %Called with stdb, returns time of last write for
  associated statement.%
                                                                        02702
     PROCEDURE (stdb);                                                  02703
     LOCAL sdbloc;                                                      02704
     lodent( stdb, sdbtyp : sdbloc);                                    02705
     RETURN ([sdbloc].stime) END.

                                                                        02706
  (getdblen) %***% %Called with stdb, returns length of data block%
                                                                        03643
     PROCEDURE (stdb);                                                  03644
     LOCAL sdbloc;                                                      03645
     lodent( stdb, sdbtyp : sdbloc);                                    03646
     RETURN ([sdbloc].slength) END.

                                                                        03647
  (getstsize) %***% %Called with STID, returns size in characters
  for associated statement.%
                                                                        02707
     PROCEDURE (stid);                                                  02708
     LOCAL sdbloc;                                                      02709
     IF NOT lodprop( stid, txttyp : sdbloc) THEN                        02710
        err($"No text block associated with this node; file probably
        bad.");                                                         02711
     RETURN ([sdbloc].schars) END.

                                                                        02712
  (fchtxt)%***% %Called with stid returns the number of the first
  character after the name of the corresponding text block .  MUST
  CHANGE CALLS ON THIS.  IF NO TEXT BLOCK, generates an error.%
                                                                        02422
     PROCEDURE (stid);                                                  02423
     LOCAL sdbloc;                                                      02424
     IF NOT lodprop (stid, txttyp : sdbloc) THEN                        02425
        err($"No text block associated with this statement; file
        probably bad.");                                                02426
     RETURN ([sdbloc].sname) END.

                                                                        02427
% Store into ring and data block fields %
                                                                        0407
  (stosuc) %Store successor of specified statement.%
                                                                        0408
     PROCEDURE (stid, sucstid);                                         0409
     LOCAL rngloc;                                                      0410
     lodent(stid, rngtyp : rngloc);                                     0411
     IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                        0412
```

```
      [rngloc].rsuc _ sucstid.stpsid;                                    0413
      RETURN END.
                                                                         0414
   (stosub) %Store sub of specified statement.%
                                                                         0415
      PROCEDURE (stid, substid);                                         0416
      LOCAL rngloc;                                                      0417
      lodent(stid, rngtyp : rngloc);                                     0418
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                         0419
      [rngloc].rsub _ substid.stpsid;                                    0420
      RETURN END.
                                                                         0421
   (stoftl) %Store tail flag of specified statement.%
                                                                         0422
      PROCEDURE (stid, tail);                                            0423
      LOCAL rngloc;                                                      0424
      lodent(stid, rngtyp : rngloc);                                     0425
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                         0426
      [rngloc].rtf _ tail;                                               0427
      RETURN END.
                                                                         0428
   (stofhd) %Store head flag of specified statement.%
                                                                         0429
      PROCEDURE (stid, head);                                            0430
      LOCAL rngloc;                                                      0431
      lodent(stid, rngtyp : rngloc);                                     0432
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                         0433
      [rngloc].rhf _ head;                                               0434
      RETURN END.
                                                                         0435
   (stonmf) %Store name flag of specified statement.%
                                                                         0436
      PROCEDURE (stid, name);                                            0437
      LOCAL rngloc;                                                      0438
      lodent(stid, rngtyp : rngloc);                                     0439
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                         0440
      [rngloc].rnamef _ name;                                            0441
      RETURN END.
                                                                         0442
   (stosdb) %Store the psdb of specified statement.%
                                                                         0443
      PROCEDURE (stid, stdb);                                            0444
      LOCAL rngloc;                                                      0445
      lodent(stid, rngtyp : rngloc);                                     0446
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                         0447
      [rngloc].rsdb _ stdb.stpsdb;                                       0448
      RETURN END.
                                                                         0449
   (stonam) %Store the name hash of specified statement.%
                                                                         0450
      PROCEDURE (stid, nameh);                                           0451
```

```
        LOCAL rngloc;                                                   0452
        lodent(stid, rngtyp : rngloc);                                  0453
        IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                        0454
        [rngloc].rnameh _ nameh;                                        0455
        RETURN END.
                                                                        0456
    (stosid) %Store the sid of specified statement.%
                                                                        0457
        PROCEDURE (stid, sid);                                          0458
        LOCAL rngloc;                                                   0459
        lodent(stid, rngtyp : rngloc);                                  0460
        IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                        0461
        [rngloc].rsid _ sid;                                            0462
        RETURN END.
                                                                        0463
    (stoorg) %***% %Store origin flag of specified statement.%
                                                                       02733
        PROCEDURE (stid, orgflg);                                      02734
        LOCAL rngloc;                                                  02735
        lodent(stid, rngtyp : rngloc);                                 02736
        IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                       02737
        [rngloc].rtorgin _ orgflg;                                     02738
        RETURN END.
                                                                       02739
    (stoitree) %***% %Store psid of inferior tree in the data block
    whose stdb is passed.%
                                                                       02791
        PROCEDURE (stdb, psid);                                        02792
        LOCAL dbloc;                                                   02793
        lodent(stdb, sdbtyp : dbloc);                                  02794
        [dbloc].sitpsid _ psid;                                        02795
        RETURN END.
                                                                       02796
% basic entity structural edits  -- core level %                       02804
  % Load entity %
                                                                         026
    (lodhdadd)  %Load header address%                                  04118
        %may create a header page%                                     04151
        %Returns                                                       04119
            (1) the CRPGAD index for the page                          04120
            (2) the address in core.%
                                                                       04121
        PROCEDURE (fileno, faddr);                                     04122
        LOCAL                                                          04123
            blkad,                                                     04129
            hdblk,    %file page number%                               04150
            pgindx,   %CRPGAD index for page%                          04125
            hd,       %pointer to header entry for page%               04126
            caddr;    %core addr to return%                            04124
        REF hd;                                                        04127
        &hd _ $rfbs + (hdblk_faddr.stblk) + filehead[fileno] - $filhed;
                                                                       04128
        IF  hdblk=0 OR hd.rfexis THEN                                  04132
```

```
        %page already exists, get it%                                   04133
        BEGIN                                                           04134
        faddr.LH _ 0;                                                   04139
        faddr.stfile _ fileno;                                          04140
        pgindx _ lodent(faddr, hdtyp : caddr);                          04141
        END                                                             04135
     ELSE                                                               04136
        %must create page%                                              04148
        BEGIN                                                           04137
        pgindx _ lodrfb(hedbas+hdblk, niltyp, fileno);                  04142
        blkad _ crpgad[pgindx];                                         04143
        [blkad].fbtype _ hdtyp;                                         04144
        [blkad].fbind _ hdblk;                                          04145
        hd.rfexis _ TRUE;                                               04146
        hd.rfused _ hd.rffree _ fbhdl;  %we don't really use this
        yet%                                                            04147
        caddr _ blkad + faddr.stwc;                                     04149
        END;                                                            04138
     RETURN(pgindx, caddr);                                             04130
     END.                                                               04131
(lodent) %***% %This routine loads file block (structural or data)
into core given its STID or STDB and the type of block, RNGTYP or
SDBTYP.  Returns                                                        02031
   (1) the CRPGAD index for the new page                                02032
   (2) the address of the sdb  or ring in core.%
                                                                        02033
   PROCEDURE (flbadr, blktyp);                                          02034
   LOCAL                                                                02035
      wc,     %word count in block to RNI or  SDB%                      02036
      pgindx, %page index where loaded%                                 02037
      tbladr,  %start of appropriate status table. %                    02038
      blkbase,   % base of block %                                      02039
      header,                                                           04166
      st;     %pointer to RNGST or DTBST entry for the page%            02040
   REF st;                                                              02041
   CASE blktyp OF                                                       02042
      = rngtyp:                                                         02043
        BEGIN                                                           02044
        tbladr _ $rngst;                                                02045
        blkbase _ rngbas;                                               02046
        END;                                                            02047
      = sdbtyp:                                                         02048
        BEGIN                                                           02049
        tbladr _ $dtbst;                                                02050
        blkbase _ dtbbas;                                               02051
        END;                                                            02052
      = hdtyp:                                                          04152
        BEGIN                                                           04153
        tbladr _ $rfbs;                                                 04155
        blkbase _ hedbas;                                               04156
        IF flbadr.stblk=0 THEN                                          04157
           %special case for 0 header page--should already be
           there%                                                       04162
           BEGIN                                                        04158
           header _ filehead[flbadr.LH];                                04160
           RETURN(header.stblk, header .A 777000B + flbadr.RH);
```

```
                                                                    04161
            END;                                                    04159
         END;                                                       04154
      ENDCASE err($"Illegal blktyp passed to lodent");              02053
   % this needs to be fast -- so we'll cheat and get the fields
   out of the flbadr without using byte pointers.  the record
   definitions for stid and stdb in utilty have a comment to the
   effect that we are doing this. %                                 02054
   r2 _ flbadr;                                                     02055
   !HLRZ r1,r2;                                                     02056
   %the stfile field is right justified in the left halfword%
                                                                    02057
   !HRRZ r2,r2;  %clear the left half%                              02058
   !IDIVI r2,1B3; %split out the block number and the word count%
                                                                    02059
   wc _ r3;  % the rngblk or dtbblk number is in r2 %               02060
   &st _ filehead[r1] + tbladr - $filhed + r2;                      02061
   %load if necessary%                                              02062
   IF (pgindx _ st.rfcore) = 0 THEN %must load it%                  02063
      pgindx _ lodrfb(r2+blkbase, blktyp, r1);                      02064
   RETURN (pgindx, crpgad[pgindx]+wc);                              02065
   END.
                                                                    02066
(lodprop) % xxx  load property %
PROCEDURE (stid, proptyp);                                          03009
   % loads the indicated property block into core.  Returns three
   items: first is FALSE if error, page number in core if success;
   second is address of block in core (which must be frozen if you
   want to do anything with it!), third is stdb of property block
   %                                                                03010
   %---------------%                                               03011
   LOCAL                                                            03012
      ptabin,                                                       03013
      stdb,                                                         03014
      blknum,                                                       03015
      db;                                                           03016
   REF db;                                                          03017
   IF stid.stastr THEN err($"String treated as statement");         03018
   % getsdb gets the first data block associated with a ring %
                                                                    03019
   IF NOT (stdb _ getsdb(stid)) THEN                                03020
      RETURN( FALSE, 0, 0);                                         03021
   ptabin _ getptab(proptyp);                                       03022
   LOOP                                                             03023
      BEGIN                                                         03024
      blknum _ lodent(stdb, sdbtyp : &db);                          03025
      IF db.sptype = proptyp THEN EXIT LOOP;                        03026
      IF  (getptab(db.sptype) > ptabin) OR (stdb.stpsdb _
      db.spsdb) = 0 THEN                                            03027
         RETURN(FALSE, 0, 0);                                       03028
      END;                                                          03029
   RETURN(blknum, &db, stdb);                                       03030
   END.                                                             03031
                                                                    03032
(lodrfb) %Load random file block.  Arguments are                      027
   (1) file block number,                                             028
```

```
(2) block type,                                                          029
   or negative number if just want to get a core page,                   030
(3) file number.                                                         031
Returns page index (i.e. the index in CRPGAD and CORPST                  032
for the page).  Look in CRPGAD indexed to get the page                   033
address.  This is NOT called to load the header of a                     034
file -- rdhdr does that job.%                                            035
PROCEDURE (nblk, btype, fileno);                                         036
LOCAL                                                                    037
   pgindex, %file block page index%                                      038
   blk,     %block address of the page%                                  039
   fl,      %file list header%                                           040
   jfn,     %file number for jsyses%                                     041
   access,  %read/write access to the page%                              042
   stent,   %pointer to status table entry%                              043
   ct;      %used as pointer into CORPST%                                044
REF ct, blk, fl, stent;                                                  045
IF nxpg NOT IN [rfpmin,rfpmax] THEN nxpg _ rfpmin;                       048
%choose the next file block page to use%                                 049
&ct _ $corpst + pgindex _ nxpg;                                          050
% look for an empty page %                                              051
WHILE ct.ctfull DO                                                       052
   BEGIN                                                                 053
   IF (pgindex _ pgindex+1) > rfpmax THEN                                054
      &ct _ $corpst + pgindex _ rfpmin                                   055
   ELSE BUMP &ct;                                                        056
   IF pgindex = nxpg THEN                                                057
      BEGIN %have checked all the pages. none empty%                     058
      %search for an unfrozen page%                                      059
      WHILE ct.ctfroz DO BEGIN                                           060
         IF (pgindex _ pgindex+1) > rfpmax THEN                          061
            &ct _ $corpst + pgindex _ rfpmin                             062
         ELSE BUMP &ct;                                                  063
         IF pgindex = nxpg THEN rerror(); %all frozen%                   064
         END;                                                            065
      IF ct.ctfile > 0 THEN %does belong to some file%                   066
         BEGIN                                                           067
         %revise the appropriate status table%                           068
         &stent _ filehead[ct.ctfile] + $rfbs - $filhed                  069
         + ct.ctpnum;                                                    070
         stent.rfcore _ 0;                                               071
         END;                                                            072
      ct.ctfull _ FALSE;                                                 073
      END;                                                               074
   END;                                                                  075
IF (nxpg _ pgindex+1) > rfpmax THEN nxpg _ rfpmin;                       076
%set up corpst for the new page%                                         077
ct _ 0;                                                                  078
ct.ctfull _ TRUE;                                                        079
IF btype < 0 THEN %just getting a page%                                  080
   BEGIN                                                                 081
   %remove the page that is there now%                                   082
   r2.LH _ 4B5;                                                          084
   r2.RH _ crpgad[pgindex] / 512;                                        085
   !pmap( -1, r2, 0);                                                    087
   RETURN (pgindex);                                                     088
```

```
      END;                                                          089
    ct.ctfile _ fileno;                                             090
    ct.ctpnum _ nblk;                                               091
    &stent _ filehead[fileno] + $rfbs - $filhed + nblk;             092
    &fl _ (fileno - 1)*filstl + $filst;                             093
    IF btype = niltyp AND fl.flpart OR stent.rfpart THEN            094
      BEGIN                                                         095
      jfn _ fl.flpart;                                              096
      access _ 14B10; %read-write%                                  097
      END                                                           098
    ELSE                                                            099
      BEGIN                                                        0100
      jfn _ fl.florig;                                             0101
      access _ 1B11; %read only%                                   0102
      END;                                                         0103
    IF jfn = 0 THEN                                                0104
      BEGIN                                                        0105
      ct _ 0;                                                      0106
      IF NOT fl.flexis THEN err($"fst entry nonexistant");         0107
      err($"Illegal JFN in LODRFB");                               0108
      END;                                                         0109
    %set up registers for PMAP%                                    0110
    &blk _ crpgad[pgindex];                                        0111
    r1.LH _ jfn;                                                   0112
    r1.RH _ nblk;                                                  0113
    r2.LH _ 4B5;                                                   0114
    r2.RH _ &blk / 512;                                            0115
    !pmap(r1, r2, access);                                         0117
    stent.rfcore _ pgindex;                                        0118
    IF btype = niltyp THEN %do the initialization%                 0119
      BEGIN                                                        0120
      IF fl.flpart THEN stent.rfpart _ TRUE;                       0121
      blk.fbpnum _ nblk;                                           0122
        %rest is done by the calling procedure%                    0123
      END                                                          0124
    ELSE %verify%                                                  0125
      IF blk.fbtype # btype OR blk.fbpnum # nblk THEN              0126
        badfil(fileno);                                            0127
    RETURN (pgindex) END.
                                                                   0128
% nodes and groups of nodes %                                    02805
  (crenod) % xxx  create node%
  PROCEDURE ( stid, rlevcnt );                                    02914
    % Gets a new ring element with no associated data blocks and
    links it into the structure at the location specified by stid
    and rlevcnt (a relative level count: < 0 is down, =0 is
    successor, > 0 is up by rlevcnt levels).  Returns stid of new
    ring or 0 if error. %                                         02915
    %---------------%                                            02916
    LOCAL newstid;                                                02917
    ON SIGNAL ELSE RETURN(0);                                     02918
    newstid _ newrng( stid.stfile );                              02919
    insgrp( stid, rlevcnt, newstid, newstid);                     02920
    RETURN(newstid);                                              02921
    END.                                                          02922
                                                                  02923
```

```
(copgrp) %***%                                                          03833
   %Given an stid as the first argument, this routine copies the
   group bounded by the second and third arguments after the first
   stid, in the direction specified by the fourth argument.  The
   fifth argument indicates whether the correspondence list should
   be updated.                                                          03834
      It proceeds by constructing new ring elements for each
      branch defined by the top-level statements in the group.
      Then it copies these data blocks into freshly allocated
      SDB's.  It then inserts the newly created group after the
      stid passed it.%                                                  03835
   %-------------%                                                      03836
   PROCEDURE(stid, dir, grp1, grp2, upctf);                             03837
   LOCAL newsid, %new stid%                                             03838
      oldsid, %old stid being processed%                                03839
      newgp1; %head of new group%                                       03840
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");             03841
   oldsid _ grp1;                                                       03842
   newsid _ newgp1 _ newrng(stid.stfile);                               03843
   LOOP                                                                 03844
      BEGIN                                                             03845
      %get stid's, this branch%                                         03846
      WHILE (oldsid := getsub(oldsid)) # oldsid DO                      03847
         newsid _ newsub(newsid);                                       03848
      %now copy property lists and point to next branch%                03849
      LOOP                                                              03850
         BEGIN                                                          03851
         copplist(oldsid, newsid);                                      03852
         IF upctf THEN upctbl(oldsid, newsid, oldsid);                  03853
         IF oldsid = grp1 THEN %branch copy complete%                   03854
            BEGIN                                                       03855
            IF oldsid = grp2 THEN %group copy complete%                 03856
               BEGIN                                                    03857
               insgrp(stid, dir, newgp1, newsid);                       03858
               RETURN(newgp1, newsid);                                  03859
               END;                                                     03860
            oldsid _ grp1 _ getsuc(oldsid);                             03861
            EXIT;                                                       03862
            END;                                                        03863
         IF NOT getftl(oldsid := getsuc(oldsid)) THEN EXIT;             03864
            %still more in plex%                                        03865
         newsid _ getsuc(newsid);                                       03866
         END;                                                           03867
      newsid _ newsuc(newsid);                                          03868
      END;                                                              03869
   END.
                                                                        03870
(copfgrp) PROCEDURE        % copy filtered group %                      03774
   (target, rlevcnt, src1, src2, vsptr);                                03775
   LOCAL retstid, newtgt, newsrc, lstlev, toplev, sqgwrk, vspec1,
   vspec2, usqcod, cacode;                                              03776
   LOCAL TEXT POINTER tp1, tp2;                                         03777
   REF sqgwrk, vsptr;                                                   03778
                                                                        03779
   % initialize %                                                       03780
      vspec1 _ vsptr.vs1;                                               03781
```

```
      vspec2 _ vsptr.vs2;                                          03782
      cacode _ vsptr.vscacode;                                     03783
      usqcod _ vsptr. vsusqcod;                                    03784
      retstid _ target;                                            03785
   &sqgwrk _ openseq (src1, src2, vspec1, vspec2, usqcod, cacode);
                                                                   03786
   ON SIGNAL ELSE IF &sqgwrk THEN                                  03787
      BEGIN                                                        03788
      ON SIGNAL ELSE;                                              03789
      closeseq( &sqgwrk := 0 );                                    03790
      END;                                                         03791
   IF (newsrc _ seqgen(&sqgwrk)) # endfil THEN                     03792
      BEGIN                                                        03793
      IF newsrc.stastr THEN                                        03794
         BEGIN                                                     03795
         FIND SF(newsrc) ^tp1 SE(newsrc) ^tp2;                     03796
         retstid _ newtgt _ cinssta( target, rlevcnt, $tp1, $tp2 )
                                                                   03797
         END                                                       03798
      ELSE retstid _ newtgt _                                      03799
         ccopsta( target, rlevcnt, newsrc, FALSE, FALSE);          03800
      toplev _ lstlev _                                            03801
         IF sqgwrk.swclvl = 0 THEN 1 ELSE sqgwrk.swclvl;           03802
      WHILE (newsrc _ seqgen(&sqgwrk)) # endfil DO                 03803
         BEGIN                                                     03804
         rlevcnt _ 0;                                              03805
         CASE sqgwrk.swclvl OF                                     03806
            =lstlev: NULL;                                         03807
            >lstlev:                                               03808
               BEGIN                                               03809
               rlevcnt _ -1;                                       03810
               lstlev _ lstlev + 1;                                03811
               END;                                                03812
            <lstlev:                                               03813
               WHILE lstlev > toplev DO                            03814
                  BEGIN                                            03815
                  BUMP rlevcnt;                                    03816
                  IF ((lstlev _ lstlev - 1) = sqgwrk.swclvl) THEN
                                                                   03817
                     EXIT LOOP 1;                                  03818
                  END;                                             03819
            ENDCASE NULL;                                          03820
         IF newsrc.stastr THEN                                     03821
            BEGIN                                                  03822
            FIND SF(newsrc) ^tp1 SE(newsrc) ^tp2;                  03823
            newtgt _ cinssta( newtgt, rlevcnt, $tp1, $tp2 )        03824
            END                                                    03825
         ELSE newtgt _                                             03826
            ccopsta( newtgt, rlevcnt, newsrc, FALSE, FALSE);       03827
         END;                                                      03828
      END;                                                         03829
   closeseq (&sqgwrk := 0);                                        03830
   RETURN(retstid, newtgt);                                        03831
   END.
                                                                   03832
(mvdlfgrp)        % move/delete/transpose filtered group %         01855
```

```
PROCEDURE                                                          01856
   (stid1,     %stid of first statement to be moved/deleted%
                                                                   01857
   stid2,      %stid of second statement to be moved deleted%
                                                                   01858
   vsptr,      %pointer to filter viewspec record%                 01859
   newstid,    %stid of statement following group: stid1 stid2%
                                                                   01860
   type,       %delete/move/transpose%                             01990
   mvdlflag,   %stid of target for move/transpose%                 01861
   mvrlev      %relative level for move/transpose%                 01862
   );                                                              01863
LOCAL                                                              01864
   vspec1, unfseq, ufstid, ufring, filseq, filstid, filring,
   cring, pflag, rstid, toplev, lstlev, mtoplev, mlstlev,
   rlevcnt, cstid, nstid, retstid;                                 01865
REF vsptr, unfseq, ufring, filseq, filring, cring;                 01866
                                                                   01867
vspec1 _ vsptr.vs1;                                                01868
   vspec1.vslev _ -1;  % all levels %                              01869
   vspec1.vscapf _ FALSE;  % no content analyzer program %
                                                                   01870
   vspec1.vsusqf _ FALSE;  % no sequence generator program %
                                                                   01871
% make unfiltered pass thru group; mark all with absolute level
%                                                                  01872
   % open an unfiltered sequence %                                01873
      &unfseq _                                                    01874
         openseq( stid1, stid2, vspec1, vsptr.vs2, FALSE,
         FALSE);                                                   01875
   ON SIGNAL ELSE IF &unfseq THEN closeseq (&unfseq :=0);          01876
   WHILE (ufstid _ seqgen(&unfseq)) # endfil DO                    01877
      BEGIN                                                        01878
      lodent( ufstid, rngtyp : &ufring);                           01879
      ufring.rinst1 _ unfseq.swclvl;                               01880
      ufring.rdummy _ TRUE;                                        01881
      END;                                                         01882
   closeseq( &unfseq := 0 );                                       01883
% make filtered pass marking all statements that pass false %
                                                                   01884
   % open a filtered sequence %                                   01885
      &filseq _                                                    01886
         openseq( stid1, stid2, vsptr.vs1, vsptr.vs2,
         vsptr.vsusqcod, vsptr.vscacode);                          01887
   ON SIGNAL ELSE IF &filseq THEN closeseq (&filseq :=0);          01888
   WHILE (filstid _ seqgen(&filseq)) # endfil DO                   01889
      BEGIN                                                        01890
      lodent(filstid, rngtyp : &filring);                          01891
      filring.rdummy _ FALSE;                                      01892
      END;                                                         01893
   closeseq( &filseq := 0 );                                       01894
   ON SIGNAL ELSE;                                                 01895
% get predessor or up (remember which) of first unfiltered %
                                                                   01896
   lodent( stid1, rngtyp : &cring);                                01897
   pflag _ IF cring.rhf THEN FALSE ELSE TRUE;                      01898
```

```
      rstid _ IF pflag THEN getprd(stid1) ELSE getup(stid1);  01899
    % remove the entire group %                                01900
      IF NOT remgrp( stid1, stid2) THEN                       01901
         CASE type OF                                         01991
            = moveflag: err($"illegal move");                 01992
            = trnsflag: err($"illegal transpose");            01993
            = deltflag: err($"illegal delete");               01994
            ENDCASE;                                          01995
    % initialize level stuff %                                01904
      toplev _ lstlev _ mtoplev _ mlstlev _ 0;                01905
    % initialize for main loop %                              01906
      cstid _ stid1;                                          01907
      retstid _ 0;                                            03726
    % final pass through removed group %                       01908
      WHILE cstid DO                                          01909
         BEGIN                                                01910
         % get ring element for current stid %                 01911
            lodent( cstid, rngtyp : &cring);                  01912
         % set up stack for next statement %                   01913
            IF cstid = stid2 THEN                             01914
               BEGIN                                          01915
               nstid _ 0;                                     01916
               !PUSH s,nstid                                  01917
               END                                            01918
            ELSE IF NOT cring.rtf THEN                        01919
               BEGIN                                          01920
               nstid _ cring.rsuc;                            01921
               nstid.stfile _ stid1.stfile;                   01922
               !PUSH s,nstid;                                 01923
               END;                                           01924
            cring.rsuc _ 0;                                   01925
            IF (nstid_ cring.rsub:= cstid.stpsid) # cstid.stpsid
            THEN                                              01926
               BEGIN                                          01927
               nstid.stfile _ stid1.stfile;                   01928
               !PUSH s,nstid;                                 01929
               END;                                           01930
         IF cring.rdummy THEN  %did not pass filter%          01931
            CASE type OF                                      01996
               = trnsflag: delgrp(cstid, cstid, newstid);     01997
               ENDCASE                                        01998
                  BEGIN   % insert the stmt back where it was %
                                                              01932
                  IF NOT toplev THEN                          01933
                     BEGIN                                    01934
                     toplev _ lstlev _ cring.rinst1;          01935
                     rlevcnt _ IF pflag THEN 0 ELSE -1;       01936
                     END                                      01937
                  ELSE rlevcnt _ 0;                           01938
                  CASE cring.rinst1 OF                        01939
                     = lstlev:  NULL;                         01940
                     > lstlev:                                01941
                        BEGIN                                 01942
                        rlevcnt _ -1;                         01943
                        BUMP lstlev;                          01944
                        END;                                  01945
```

```
                        < lstlev:                                       01946
                           WHILE lstlev > toplev DO                     01947
                              BEGIN                                     01948
                              BUMP rlevcnt;                             01949
                              IF ((lstlev _ lstlev - 1) =
                              cring.rinst1) THEN                        01950
                                 EXIT LOOP 1;                           01951
                              END;                                      01952
                        ENDCASE;                                        01953
                     insgrp( rstid := cstid, rlevcnt, cstid, cstid );
                                                                        01954
                     IF (type = deltflag) AND NOT retstid THEN
                     retstid _ cstid;                                   03727
                     END                                                01955
               ELSE   %did pass the filter%                             01956
                  CASE type OF                                          02000
                     = deltflag: delgrp(cstid, cstid, newstid);         02001
                     ENDCASE                                            02002
                        BEGIN   % move the statement %                  01959
                        IF NOT mtoplev THEN                             01960
                           BEGIN                                        01961
                           mtoplev _ mlstlev _ cring.rinst1;            01962
                           rlevcnt _ mvrlev;                            01963
                           END                                          01964
                        ELSE rlevcnt _ 0;                               01965
                        CASE cring.rinst1 OF                            01966
                           = mlstlev:  NULL;                            01967
                           > mlstlev:                                   01968
                              BEGIN                                     01969
                              rlevcnt _ -1;                             01970
                              BUMP mlstlev;                             01971
                              END;                                      01972
                           < mlstlev:                                   01973
                              WHILE mlstlev > mtoplev DO                01974
                                 BEGIN                                  01975
                                 BUMP rlevcnt;                          01976
                                 IF ((mlstlev _ mlstlev - 1) =
                                 cring.rinst1) THEN                     01977
                                    EXIT LOOP 1;                        01978
                                 END;                                   01979
                           ENDCASE;                                     01980
                        insgrp( mvdlflag := cstid, rlevcnt, cstid, cstid
                        );                                              01981
                        IF NOT retstid THEN retstid _ cstid;            03728
                        END;                                            01982
               % get next stid %                                        01983
                  !POP s,cstid;                                         01984
               END;                                                     01985
         IF NOT retstid THEN                                            03729
            CASE type OF                                                03730
               = deltflag:  retstid _ newstid;                          03731
               ENDCASE   retstid _ stid1;                               03732
         RETURN (retstid);                                              01986
         END.                                                           01987
(remgrp)                                                                01230
   %Removes group whose bounds are passed it from position in ring
```

```
   but does not delete either ring elements or SDB's.%                01231
   %-------------%                                                    01232
   PROCEDURE(grp1, grp2);                                             01233
   LOCAL stid; %work area for updating stid%                          01234
   IF grp1.stpsid = origin OR                                         01235
      grp2.stpsid = origin THEN RETURN(FALSE);                        01236
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");           01237
   %putcdpanic(IF getfhd(grp1) THEN getup(grp1)                       01238
      ELSE getprd(grp1));%                                            01239
   IF getfhd(grp1) THEN                                               01240
      BEGIN                                                           01241
      stid _ getsuc(grp2); %new plex head%                            01242
      IF NOT getftl(grp2) THEN stofhd(stid, TRUE);                    01243
      stosub(getup(grp2), stid);                                      01244
      END                                                             01245
   ELSE                                                               01246
      BEGIN                                                           01247
      stid _ getprd(grp1);                                            01248
      stoftl(stid, getftl(grp2));                                     01249
      stosuc(stid, getsuc(grp2));                                     01250
      END;                                                            01251
   RETURN;                                                            01252
   END.
                                                                      01253
(delgrp) %***%                                                        02396
   %This routine destroys all trace of the group bounded by the
   arguments passed it.  (Note that it does not remove the group
   from the ring; it assumes that this has been done by, for
   example, REMGRP.)                                                  02397
      It follows each branch to its deepest level, deleting
      references to substructure in the sub fields; it then
      follows the structure back up, through the suc pointers,
      freeing SDB's as it goes.%                                      02398
   %-------------%                                                    02399
   PROCEDURE(grp1, grp2, newsid);                                     02400
   LOCAL                                                              02401
      stid, %stid being processed%                                    02402
      subsid; %sub of stid being processed%                           02403
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");           02404
   IF grp1.stpsid = origin OR                                         02405
      grp2.stpsid = origin THEN err($"illegal delete");               02406
   stid _ grp1;                                                       02407
   LOOP                                                               02408
      BEGIN                                                           02409
      WHILE (subsid _ getsub(stid)) # stid DO                         02410
         BEGIN                                                        02411
         stosub(stid, stid);                                          02412
         stid _ subsid;                                               02413
         END;                                                         02414
      stid _ getsuc(subsid); %now go back up%                         02415
      upctbl(subsid, endfil, newsid);                                 02416
      freplist(subsid); %free property list%                          02417
      frerng(subsid);                                                 02418
      IF subsid = grp2 THEN RETURN;                                   02419
      END;                                                            02420
   END.
```

```
                                                                    02421
% property list %                                                   02806
   (copplist) % xxx ***% %Copies a property llst.  Arguments are
                                                                    02817
      (1) source STID,                                              02818
      (2) destination STID.                                         02819
      It is assumed that destination does NOT have an SDB currently
      associated with it.%
                                                                    02820
      PROCEDURE (source, dest);                                     02821
      LOCAL                                                         02822
         db, % address in core of current data block %              02823
         oldb; % stdb of current data block %                       02824
      REF db;                                                       02825
      % find stdb of first property block to be copied.  returns 0 if
      origin of inferior tree. %                                    02826
         IF NOT (oldb _ getsdb(source)) THEN RETURN;                02827
      LOOP                                                          02828
         BEGIN                                                      02829
         % load the current property block into core to get its type
         %                                                          02830
            lodent( oldb, sdbtyp : &db );                           02831
         % Copy the property; SIGNAL if error occurs %             02832
            IF NOT (copprop( source, db.sptype, dest)) THEN         02833
               err($"File system error on copy.  File may be bad.");
                                                                    02834
         % get the next property  stdb: returns false if no more in
         list %                                                     02835
            IF NOT (oldb _ getpsdb(oldb)) THEN EXIT LOOP;           02836
         END;                                                       02837
      RETURN END.
                                                                    02838
   (freplist) %***% PROCEDURE (stid);                               02188
      % free all properties associated with this node.  calls freprop
      until the last property is reached %                          02189
      LOCAL                                                         02190
         rnl,                                                       03710
         sdb,                                                       03711
         next,                                                      03712
         fileno,                                                    03713
         sdbit; % stid of inferior tree of property freed; it must be
         freed as well %                                            03714
      REF rnl, sdb;                                                 02191
      lodent(stid, rngtyp : &rnl);                                  02192
      IF rnl.rtorgin THEN RETURN;                                   02193
      IF (next _ rnl.rsdb) = 0 THEN RETURN;                         02194
      fileno _ stid.stfile;                                         02195
      LOOP                                                          02196
         BEGIN                                                      02197
         next.stfile _ fileno;                                      02198
         lodent(next, sdbtyp : &sdb);                               02199
         IF sdb.sptype = txttyp THEN                                02200
            BEGIN                                                   02201
            % reset this string in correspondence table before
            getting rid of it %                                     02202
            FIND SF(stid) ^sptr1 SE(stid) ^sptr2;                   02203
```

```
            cldsr ($sptr1);                                          02204
            END;                                                     02205
         IF (sdbit _ freprop( next := sdb.spsdb)) THEN               02206
            freintree( sdbit );                                      03715
         IF NOT next THEN EXIT LOOP;                                 02207
         END;                                                        02208
      stosdb( stid, 0);                                              02209
      RETURN;                                                        02210
      END.

                                                                     02211
% property %                                                         02807
   (creprop) % xxx   Create property block %
   PROCEDURE ( stid, proptyp, length, data );                        03390
      % Builds a data block of property type proptyp (which must be a
      valid type atssigned (and declared ) by ARC and links it into
      the plist associated with the stid in the proper order
      (determined in the procedure locprop).  If such a property
      already exists in the node, we have an error:  it must first be
      deleted.  Returns stdb of new block or 0 if error.  length is
      the length of the data and data is a pointer to an array of
      length words in which the data is stored.  If &data is zero, do
      not copy data; just initialize the block. Calls crepr2 to do
      actual work. %                                                 03391
      %---------------%                                             03392
      RETURN( crepr2( stid, proptyp, length, data, FALSE, 0, 0) );
                                                                     03393
      END.                                                           03394
                                                                     03395
   (copprop) %xxx copy property%
   PROCEDURE ( stid, % of source %                                   02839
      proptyp,                                                       02840
      destid % of destination node % );                              02841
      % copies property block (and associated inferior tree if any)
      from block indicated by stid and proptyp to a new block to be
      created on destid.  Returns TRUE if OK. 0 if error %          02842
      %---------------%                                             02843
      LOCAL                                                          02844
         sdbblk, % page number with source sdb %                     02845
         sdbold, % address of source sdb %                           02846
         stdb, % stdb of source sdb %                                02847
         destblk, % page number with destination (ring or sdb) %
                                                                     02848
         desel, % address of destination ring or sdb %               02849
         dest, %stdb or stpsid of destination %                      02850
         destrng, % TRUE if destination a ring, FALSE if a property %
                                                                     02851
         nwblk, % page with new db %                                 02852
         sdbnew, % address of new db %                               02853
         nwtdb, % stdb of new sdb %                                  02854
         room, % size of source db %                                 02855
         intre; % stid of origin of inferior tree %                  02856
      REF sdbold, desel, sdbnew;                                     02857
      % Get and freeze source block %                               02858
         ON SIGNAL                                                   02859
            = errsig:                                                03738
               BEGIN                                                 03739
```

```
            dismes(2, sysmsg);                                          03740
            RETURN( FALSE );                                            03741
            END;                                                        03742
         ELSE RETURN( FALSE );                                          03737
      IF NOT (sdbblk _ lodprop(stid, proptyp : &sdbold, stdb ))
      THEN                                                              02860
         RETURN( FALSE );                                               02861
      frzblk( sdbblk, 1);                                               02862
      ON SIGNAL                                                         02863
         = errsig:                                                      03744
            BEGIN                                                       03745
            dismes(2, sysmsg);                                          03746
            frzblk(sdbblk, -1);                                         03749
            RETURN( FALSE );                                            03747
            END;                                                        03748
         ELSE                                                           03743
            BEGIN                                                       02864
            frzblk(sdbblk, -1);                                         02865
            RETURN(FALSE);                                              02866
            END;                                                        02867
   % Get and freeze destination; locprop finds the proper location
   for this new block. %                                                02868
      IF NOT (destblk _ locprop(destid, proptyp : &desel, dest,
      destrng)) THEN                                                    02869
         BEGIN                                                          02870
         frzblk(sdbblk, -1);                                            02871
         RETURN(FALSE);                                                 02872
         END;                                                           02873
      frzblk( destblk, 1);                                              02874
      ON SIGNAL                                                         02875
         =errsig:                                                       03751
            BEGIN                                                       03752
            dismes(2, sysmsg);                                          03757
            frzblk(sdbblk, -1);                                         03753
            frzblk(destblk, -1);                                        03754
            RETURN(FALSE);                                              03755
            END;                                                        03756
         ELSE                                                           03750
            BEGIN                                                       02876
            frzblk(sdbblk, -1);                                         02877
            frzblk(destblk, -1);                                        02878
            RETURN(FALSE);                                              02879
            END;                                                        02880
   % Get a new block %                                                  02881
      room _ sdbold.slength;                                            02882
      nwtdb _ newdb(room, proptyp, dest.stfile : &sdbnew, nwblk);
                                                                        02883
      frzblk( nwblk, 1);                                                02884
      ON SIGNAL                                                         02885
         = errsig:                                                      03759
            BEGIN                                                       03760
            dismes(2, sysmsg);                                          03766
            frzblk(sdbblk, -1);                                         03761
            frzblk(destblk, -1);                                        03762
            frzblk(nwblk, -1);                                          03763
            RETURN(FALSE);                                              03764
```

```
            END;                                                    03765
         ELSE                                                       03758
            BEGIN                                                   02886
            frzblk(sdbblk, -1);                                     02887
            frzblk(destblk, -1);                                    02888
            frzblk(nwblk, -1);                                      02889
            RETURN(FALSE);                                          02890
            END;                                                    02891
      % copy data from source to new block %                        02892
         mvbfbf( &sdbold, &sdbnew, room);                           02893
      % copy name fields from old block to new %                    02894
         copnam( destrng, &sdbold, stid, &desel, dest);             02895
      % link in the property. lnkprop assumes appropriate blocks are
      in core and frozen.  locprop has been used to find the correct
      location for the new property block %                         02896
         lnkprop( destrng, &desel, dest, &sdbnew, nwtdb);           02897
      % Unfreeze destination and source %                           02898
         frzblk(sdbblk, -1);                                        02899
         frzblk(destblk, -1);                                       02900
         ON SIGNAL                                                  02902
            =errsig:                                                03768
            BEGIN                                                   03769
            dismes(2, sysmsg);                                      03773
            frzblk(nwblk, -1);                                      03770
            RETURN(FALSE);                                          03771
            END;                                                    03772
         ELSE                                                       03767
            BEGIN                                                   03734
            frzblk(nwblk, -1);                                      02901
            RETURN(FALSE);                                          03733
            END;                                                    03735
      % copy inferior tree if any.  Put the copy's STPSID in the
      SITPSID field of the new block. %                             02903
         IF sdbnew.sitpsid THEN                                     02904
            BEGIN                                                   02905
            intre _ sdbnew.sitpsid;                                 02906
            intre.stfile _ stid.stfile;                             02907
            IF NOT trecop( intre, nwtdb) THEN RETURN( FALSE );      02908
            % trecop also links the tree to nwtdb %                 02909
            END;                                                    02910
      frzblk(nwblk, -1);                                            03736
      RETURN(nwtdb);                                                02911
      END.                                                          02912
                                                                    02913

   (movprop) % xxx  move property %
   PROCEDURE ( stid, % of source %                                  03323
      proptyp,                                                      03324
      destid % of destination node % );                             03325
      % Moves the property indicated from node specifed by stid to
      node specified by destid.  Accomplishes this by unlinking and
      relinking the block.  If a property type of the type being
      moved exists at the destination, we have an error.  Returns
      true if OK, 0 if error. %                                     03326
      %---------------%                                            03327
      LOCAL                                                         03328
         sdbblk, % page number with source sdb %                    03329
```

```
      sdbold, % address of source sdb %                                 03330
      stdb, % stdb of source sdb %                                      03331
      destblk, % page number with destination (ring or sdb) %
                                                                        03332
      desel, % address of destination ring or sdb %                     03333
      dest, %stdb or stpsid of destination %                            03334
      destrng; % TRUE if destination a ring, FALSE if a property %
                                                                        03335
   REF sdbold, desel;                                                   03336
   % If two files involved, copy property to new node. %                03337
      IF stid.stfile # destid.stfile THEN                               03338
         BEGIN                                                          03339
         % ---- INTERFILE MOVES ---- %                                 03340
         % Copy block to new file; copprop links it in %                03341
            IF NOT (copprop(stid, proptyp, destid )) THEN               03342
               RETURN( FALSE );                                         03343
         % delete the original %                                        03344
            IF NOT( delprop( stid, proptyp )) THEN                      03345
               RETURN( FALSE );                                         03346
         RETURN( TRUE );                                                03347
         END;                                                           03348
   % ---- INTRAFILE MOVES ---- %                                       03349
   % Get and freeze source block %                                      03350
      ON SIGNAL ELSE RETURN( FALSE );                                   03351
      IF NOT (sdbblk _ lodprop(stid, proptyp : &sdbold, stdb ))
      THEN                                                              03352
         RETURN( FALSE );                                               03353
      frzblk( sdbblk, 1);                                               03354
      ON SIGNAL ELSE                                                    03355
         BEGIN                                                          03356
         frzblk(sdbblk, -1);                                            03357
         RETURN(FALSE);                                                 03358
         END;                                                           03359
   % Get and freeze destination; locprop finds the proper location
   for this new block. %                                                03360
      IF NOT (destblk _ locprop(destid, proptyp : &desel, dest,
      destrng)) THEN                                                    03361
         BEGIN                                                          03362
         frzblk(sdbblk, -1);                                            03363
         RETURN(FALSE);                                                 03364
         END;                                                           03365
      frzblk( destblk, 1);                                              03366
      ON SIGNAL ELSE                                                    03367
         BEGIN                                                          03368
         frzblk(sdbblk, -1);                                            03369
         frzblk(destblk, -1);                                           03370
         RETURN(FALSE);                                                 03371
         END;                                                           03372
   % copy name fields from old to new ring %                           03373
      copnam( destrng, &sdbold, stid, &desel, dest);                    03374
   % Unlink current property location; relink nodes around it. %
                                                                        03375
      IF NOT (unlnkprop( &sdbold,  stdb)) THEN                          03376
         BEGIN                                                          03377
         frzblk(sdbblk, -1);                                            03378
         frzblk(destblk, -1);                                           03379
```

```
            RETURN(FALSE);                                          03380
            END;                                                    03381
      % link in the property. lnkprop assumes appropriate blocks are
      in core and frozen.  locprop has been used to find the correct
      location for the new property block %                         03382
         lnkprop( destrng, &desel, dest, &sdbold, stdb);            03383
      % Unfreeze destination and source %                           03384
         frzblk(sdbblk, -1);                                        03385
         frzblk(destblk, -1);                                       03386
      RETURN( TRUE );                                               03387
      END.                                                          03388
                                                                    03389

   (delprop) %xxx delete property %
   PROCEDURE ( stid, proptyp );                                     02936
      % deletes the property block and any associated inferior tree
      structure  for the block proptyp block of the indicated node.
      Returns TRUE if successful, 0 if not. %                       02937
      %---------------%                                            02938
      LOCAL sdb, stdb;                                              02939
      LOCAL sdbit; % stid of inferior tree of property freed; it must
      be freed as well %                                            03716
      REF sdb;                                                      02940
      ON SIGNAL ELSE RETURN(0);                                     02941
      IF NOT lodprop( stid, proptyp : &sdb, stdb) THEN              02942
         RETURN(0);                                                 02943
      IF (sdbit _ freprop( stdb)) THEN                              03717
         freintree( sdbit );                                        03718
      RETURN( TRUE );                                               02945
      END.                                                          02946
                                                                    02947

   (reprop) % xxx  replace property %
   PROCEDURE (stid, proptyp, length,  data);                        03537
      % replaces the property block indicated by stid and proptyp
      with a block with data as indicated.  If length is the same as
      the length of data in existing property block, a short cut may
      be taken and the data overwrites the old data.  If, however,
      the length is different, a new block is built and linked in.
      The inferior tree is not replaced in any case:  it remains the
      same.  The inferior tree's pointer to the "owning" property
      block is changed.  Uses filesc if this is a text block.  %
                                                                    03538
      %---------------%                                            03539
      LOCAL                                                         03540
         sdb,                                                       03541
         sdbblk,                                                    03542
         stdb,                                                      03543
         sdbit; % stid of inferior tree of freed property; will be
         linked into new property %                                 03719
      REF data, sdb;                                                03547
      ON SIGNAL ELSE RETURN( FALSE );                               03548
      IF proptyp = txttyp THEN                                      03549
         BEGIN                                                      03550
         *sar* _ *[&data-1]*;                                       03551
         rplsid _ stid;                                             03552
         filesc();                                                  03553
         RETURN( TRUE );                                            03554
```

```
      END;                                                          03555
    % get the block to be replaced.  If there is none, this is an
    error %                                                         03556
      IF NOT (sdbblk _ lodprop( stid, proptyp : &sdb, stdb)) THEN
                                                                    03557
        RETURN(FALSE);                                              03558
    % if the length is different, delete the block and create a new
    one.  Otherwise, simply copy the data into the correct
    location.  (Since text handled elsewhere this simplification
    works! %                                                        03559
      IF (sdb.slength-sdbhdl) = length THEN                         03560
        BEGIN                                                       03561
        frzblk( sdbblk, 1);                                         03577
        ON SIGNAL ELSE                                              03578
          BEGIN                                                     03579
          frzblk(sdbblk, -1);                                       03580
          RETURN( FALSE );                                          03581
          END;                                                      03582
        sdb.sinit _ cinit;                                          03583
        sdb.stime _ gtadcall();                                     03585
        IF &data THEN mvbfbf( &data, &sdb+sdbhdl, length);          03586
        frzblk( sdbblk, -1);                                        03587
        RETURN( TRUE );                                             03588
        END                                                         03589
      ELSE                                                          03883
        BEGIN                                                       03884
        sdbit _ freprop( stdb );                                    03885
        IF (stdb _ crepr2( stid, proptyp, length, &data, FALSE,
        0, 0)) THEN                                                 03886
          BEGIN                                                     03887
          % link old inferior tree to new data block %              03888
            IF sdbit THEN insitree( sdbit, stdb );                  03889
          END;                                                      03890
        RETURN( stdb % TRUE if new block, FALSE if not % );         03891
        END;                                                        03892
    END.

                                                                    03595
% Inferior tree %                                                  02808
  (creit) %xxx create inferior tree %
  PROCEDURE (stid, proptyp);                                        02924
    % Creates the origin of an inferior tree and links it to the
    data block property specified by stid and proptyp.  Returns 0
    if error or stid of origin of inferior tree. %                  02925
    %---------------%                                              02926
    LOCAL sdb, stdb, newstid, ring;                                 02927
    REF sdb, ring;                                                  02928
    IF NOT lodprop (stid, proptyp : &sdb, stdb) THEN                02929
      RETURN(0);                                                    02930
    newstid _ newrng( stid.stfile: &ring);                          02931
    % Initialize inferior tree origin fields %                     03678
      ring.rsuc _ ring.rsub _ newstid.stpsid;                       03679
      ring.rhf _ ring.rtf _ ring.rtorgin _ TRUE;                    03680
      % rnameh and rnamef and rsid have been set in newrng.  Other
      fields were also zeroed. %                                    03681
    insitree(newstid, stdb);                                        02932
    RETURN(newstid);                                                02933
```

```
   END.                                                          02934
                                                                 02935
(copit) % xxx copy inferior tree%
PROCEDURE ( stid, proptyp, destid );                             02992
   % copies inferior tree of property block at node indicated by
   stid and proptyp to the proptyp block of destid.  Returns TRUE
   if successful, 0 if error %                                   02993
   %---------------%                                            02994
   LOCAL sdb, stdb, stidit;                                      02995
   REF sdb;                                                      02996
   ON SIGNAL ELSE RETURN( FALSE );                               02997
   IF NOT lodprop(stid, proptyp : &sdb, stdb ) THEN              02998
      RETURN( FALSE );                                           02999
   IF NOT stidit _ sdb.sitpsid THEN                              03000
      RETURN( TRUE );                                            03001
   stidit.stfile _ stid.stfile;                                  03002
   IF NOT lodprop(destid, proptyp : &sdb, stdb ) THEN            03003
      RETURN( FALSE );                                           03004
   IF NOT trecop( stidit, stdb ) THEN RETURN( FALSE );           03005
   RETURN( TRUE );                                               03006
   END.                                                          03007
                                                                 03008
(movit) % xxx move inferior tree%
PROCEDURE ( stid, proptyp, destid);                              02963
   % moves the inferior tree associated with property block
   indicated by stid and  proptyp to  the property block proptyp
   associated with node destid  Returns true if OK, 0 if error %
                                                                 02964
   %---------------%                                            02965
   LOCAL sdb, stdb, stidit;                                      02966
   REF sdb;                                                      02967
   ON SIGNAL ELSE RETURN( FALSE );                               02968
   IF NOT lodprop( stid, proptyp: &sdb, stdb ) THEN              02969
      RETURN( FALSE );                                           02970
   IF NOT (stidit _ sdb.sitpsid := 0) THEN                       02971
      RETURN( TRUE ); % No inferior tree to move %               02972
   stidit.stfile _ stid.stfile;                                  02973
   IF NOT lodprop( destid, proptyp: &sdb, stdb ) THEN            02974
      BEGIN                                                      02975
      % release the inferior tree %                             02976
         freintree( stidit );                                    02977
      RETURN( FALSE );                                           02978
      END;                                                       02979
   IF stid.stfile = destid.stfile THEN                           02980
      insitree( stidit, stdb )                                   02981
   ELSE                                                          02982
      BEGIN                                                      02983
      % Copy & link in inferior tree to new file %              02984
         IF NOT trecop( stid, stdb ) THEN RETURN( FALSE );       02985
      % Release the old inferior tree %                         02986
         freintree( stidit );                                    02987
      END;                                                       02988
   RETURN( TRUE );                                               02989
   END.                                                          02990
                                                                 02991
(delit) %xxx delete inferior tree%
```

```
    PROCEDURE (stid, proptyp );                                          02948
      % Deletes the inferior tree of the indicated property block.
      Unlinks it and releases space. Returns True if successful, 0 if
      not. %                                                             02949
      %---------------%                                                  02950
      LOCAL sdb, stdb, stidit;                                           02951
      REF sdb;                                                           02952
      ON SIGNAL ELSE RETURN(0);                                          02953
      IF NOT lodprop( stid, proptyp : &sdb, stdb) THEN                   02954
         RETURN(0);                                                      02955
      IF NOT (stidit _ sdb.sitpsid) THEN                                 02956
         RETURN(TRUE) % none to delete %                                 02957
      ELSE stidit.stfile _ stid.stfile;                                  02958
      freintree( stidit );                                               02959
      RETURN( TRUE );                                                    02960
      END.                                                               02961
                                                                         02962
% basic entity structural edit support %                                02809
   % edit support %                                                      03597
      (freprop) % xxx ***% PROCEDURE (stdb);                             03251
         % Frees property, but NOT associated inferior tree!  The
         calling procedure must explicitly call freintree with the stid
         of the  inferior tree to have that done.  (This permits
         replacement of data without removing the inferior tree.)  To
         facilitate that, this procedure returns: FALSE if there is no
         inferior tree or the stid of the inferior tree if there is one.
         %                                                               03725
         LOCAL                                                           03252
            blknum, % index into CRPGAD fo this block %                  03253
            blkad,  %address of block containing the SDB%                03254
            sdbit, % psid of inferior tree; if non-zero, we must release
            the inferior tree including all its structure and data
            elements. %                                                  03255
            dt,    %pointer to DTBST entry for the block%                03256
            blkfre, %address of free space for the block%                03257
            sdbblk, %address of SDB to be freed%                         03258
            sdbpt,  %pointer to SDB's in the block%                      03259
            savstdb, % stdb of first db in block of stdb (stwc field
            zeroed) %                                                    03627
            fileno, % of file in which stdb lives %                      03628
            blknxt; %another pointer to SDB's in the block%              03261
         REF dt;                                                         03262
         IF stdb.stpsdb = 0 THEN RETURN( FALSE );                        03263
         % get block to be freed into core; sdbblk will be its address
         in core and blkad will be the address of the start of the page
         in which it lives. %                                            03264
            % Set stwc field of savstdb copy of stdb  to zero so that
            lodsdb  will return as its second parameter the address in
            core of the first word in the page containing the data
            block.  Get the address  of the data block by adding wc back
            in. %                                                        03265
            fileno _ stdb.stfile;                                        03629
            savstdb _ stdb;                                              03630
            savstdb.stwc _ 0;                                            03631
            dblst _ stdb.stblk; % save in global for use by newdb %
                                                                         03267
```

```
   blknum _ lodent(savstdb, sdbtyp :blkad);                          03268
   frzblk( blknum, 1);                                               03269
   ON SIGNAL ELSE frzblk(blknum,-1);                                 03270
   sdbblk _ blkad + stdb.stwc;                                       03271
% save away  pointer to inferior tree.  we must delete it as
well.  We do it after this block is unfrozen below. %                03272
   IF (sdbit _ [sdbblk].sitpsid) THEN                                03273
      sdbit.stfile _ fileno;                                         03274
% Make PC if necessary; calls err if trouble, resets if
catastrophe %                                                        03598
   crepc(fileno);                                                    03599
% Unlink the block  Does not delete inferior tree.%                  03275
    IF NOT unlnkprop( sdbblk, stdb) THEN                             03276
      BEGIN                                                          03277
      err($"System error while unlinking data block.  Possible
      bad file.");                                                   03600
      END;                                                           03281
%pointer to the dtbst entry%                                         03282
   &dt _ filehead[fileno] + $dtbst - $filhed + stdb.stblk;
                                                                     03283
%record decrease in used count%                                      03284
   dt.rfused _ dt.rfused - [sdbblk].slength;                         03285
%pointer to free space%                                              03286
   % get absolute location of start of free block for this page
   %                                                                 03287
   blkfre _ dt.rffree + blkad;                                       03288
   % get absolute core address of first sdb in page %               03289
   sdbpt _ blkad + fbhdl;                                            03290
   IF sdbblk > sdbpt THEN                                            03291
      %the block to be freed is not the first%                       03292
      %merge the preceeding SDB if it is garbage%                    03293
      BEGIN                                                          03294
      %move sdbpt to the SDB in front of the one to be               03295
      freed%                                                         03296
      LOOP CASE blknxt _ sdbpt + [sdbpt].slength OF                  03297
         = sdbblk : EXIT;                                            03298
         <= sdbpt : badfil(fileno);                                  03299
         ENDCASE sdbpt _ blknxt;                                     03300
      %if it is garbage, then merge%                                 03301
      IF [sdbpt].sgarb THEN                                          03302
         BEGIN                                                       03303
         [sdbpt].slength _ [sdbpt].slength +                         03304
         [sdbblk].slength;                                           03305
         sdbblk _ sdbpt;                                             03306
         END;                                                        03307
      END;                                                           03308
   [sdbblk].sgarb _ TRUE;                                            03309
   IF (blknxt _ [sdbblk].slength + sdbblk) = blkfre THEN             03310
      %can add to free space%                                        03311
      dt.rffree _ sdbblk - blkad                                     03312
   ELSE IF [blknxt].sgarb THEN                                       03313
      %merge the two%                                                03314
      [sdbblk].slength _ [sdbblk].slength +                          03315
      [blknxt].slength;                                              03316
% Unfreeze frozen block %                                            03317
   frzblk(blknum, -1);                                               03318
```

```
      ON SIGNAL ELSE;                                                03632
   RETURN( sdbit );  % the inferior tree stid: it has not been
   released! %                                                       03321
   END.
                                                                     03322
(crepr2) % xxx   Create property block %
PROCEDURE ( stid, proptyp, length, data, initflg, dlleft, dlright
);                                                                   03396
   % Builds a data block of property type proptyp (which must be a
   valid type atssigned (and declared ) by ARC and links it into
   the plist associated with the stid in the proper order
   (determined in the procedure locprop).  If such a property
   already exists in the node, we have an error:  it must first be
   deleted.  Returns stdb of new block or 0 if error.  length is
   the length of the data and data is a pointer to an array of
   length words in which the data is stored. If initflg is TRUE,
   use the values in dlleft, dlright for the name delimiters;
   otherwise compute them if this is a text block.  If &data is
   zero, do not copy data; just initialize the block. %              03397
   %---------------%                                                 03398
   LOCAL                                                             03399
      destblk, % page number with destination (ring or sdb) %
                                                                     03400
      desel, % address of destination ring or sdb %                  03401
      dest, %stdb or stid of destination %                           03402
      destrng, % TRUE if destination a ring, FALSE if a property %
                                                                     03403
      nwblk, % page with new db %                                    03404
      sdbnew, % address of new db %                                  03405
      nwtdb, % stdb of new sdb %                                     03406
      fhdloc, % address of the file header %                         03407
      rnl, % address of the ring element %                           03408
      upstid; % stid of up used in calculating name delimiters %
                                                                     03641
   LOCAL STRING escname[100];                                        03409
   REF data, desel, sdbnew, rnl;                                     03410
   % Get and freeze destination; locprop finds the proper location
   for this new block. %                                             03411
      IF NOT (destblk _ locprop(stid, proptyp : &desel, dest,
      destrng)) THEN                                                 03412
         BEGIN                                                       03413
         RETURN(FALSE);                                              03414
         END;                                                        03415
      frzblk( destblk, 1);                                           03416
      ON SIGNAL ELSE                                                 03417
         BEGIN                                                       03418
         frzblk(destblk, -1);                                        03419
         RETURN(FALSE);                                              03420
         END;                                                        03421
   % Get a new block %                                               03422
      nwtdb _ newdb(length+sdbhdl, proptyp, dest.stfile : &sdbnew,
      nwblk);                                                        03423
      frzblk( nwblk, 1);                                             03424
      ON SIGNAL ELSE                                                 03425
         BEGIN                                                       03426
         frzblk(destblk, -1);                                        03427
```

```
        frzblk(nwblk, -1);                                          03428
        RETURN(FALSE);                                              03429
        END;                                                        03430
    % copy data from buffer to new block if requested %             03431
      IF &data THEN mvbfbf( &data, &sdbnew+sdbhdl, length);         03432
    % link in the property. lnkprop assumes appropriate blocks are
    in core and frozen.  locprop has been used to find the correct
    location for the new property block %                           03433
      lnkprop( destrng, &desel, dest, &sdbnew, nwtdb);              03434
    % set name and other text property dependent fields %           03435
      IF proptyp = txttyp THEN                                      03436
        BEGIN                                                       03437
        % Initialize the name delimiters to be assigned to the
        statement. %                                                03438
          IF NOT initflg THEN                                       03439
            BEGIN                                                   03440
            % Must calculate the delimiters %                       03441
            IF stid.stpsid = origin THEN                            03442
              BEGIN %use standard delimiters for that file%
                                                                    03443
              fhdloc _ filehead[stid.stfile] - $filhed;             03444
              sdbnew.slnmdl _ [fhdloc + $namdl1];                   03445
              sdbnew.srnmdl _ [fhdloc + $namdl2];                   03446
              END                                                   03447
            ELSE                                                    03448
              BEGIN %use same delimiters as up unless up is
              origin of inferior tree%                              03449
              upstid _ getup( stid );                               03633
              IF getorf( upstid ) THEN                              03634
                BEGIN %use standard delimiters for that file%
                                                                    03635
                fhdloc _ filehead[stid.stfile] - $filhed;
                                                                    03636
                sdbnew.slnmdl _ [fhdloc + $namdl1];                 03637
                sdbnew.srnmdl _ [fhdloc + $namdl2];                 03638
                END                                                 03639
              ELSE                                                  03640
                sdbnew.slnmdl _ getnmdl( upstid :
                sdbnew.srnmdl);                                     03450
              END;                                                  03451
            END                                                     03452
          ELSE                                                      03453
            BEGIN                                                   03454
            % Use the old values passed as parameters. %            03455
            sdbnew.slnmdl _ dlleft;                                 03456
            sdbnew.srnmdl _ dlright;                                03457
            END;                                                    03458
        sdbnew.schars _ [&data-1].L; % assume data non zero for
        text! %                                                     03459
        % check for name %                                          03460
          stcwrk _ stid;                                            03461
          stcwr1 _ 1;                                               03462
          fechc1( forward, $stcwrk );                               03463
          xtrnam( $escname, $stcwrk, sdbnew.slnmdl,
          sdbnew.srnmdl );                                          03464
        sdbnew.sname _ IF escname.L = empty THEN 1 ELSE stcwr1;
```

```
                                                                    03465
        IF destrng THEN                                             03466
           BEGIN                                                    03467
           IF escname.L = empty THEN                                03468
              BEGIN                                                 03469
              desel.rnamef _ desel.rnameh _ 0;                      03470
              END                                                   03471
           ELSE                                                     03472
              BEGIN                                                 03473
              desel.rnameh _ hash($escname);                        03474
              desel.rnamef _ TRUE;                                  03475
              END;                                                  03476
           END                                                      03477
        ELSE                                                        03478
           BEGIN                                                    03479
           lodent(  stid, rngtyp : &rnl );                          03480
           IF escname.L = empty THEN                                03481
              BEGIN                                                 03482
              rnl.rnamef _ rnl.rnameh _ 0;                          03483
              END                                                   03484
           ELSE                                                     03485
              BEGIN                                                 03486
              rnl.rnameh _ hash($escname);                          03487
              rnl.rnamef _ TRUE;                                    03488
              END;                                                  03489
           END;                                                     03490
        END;                                                        03491
     % Unfreeze blocks %                                           03492
        frzblk(destblk, -1);                                        03493
        frzblk(nwblk, -1);                                          03494
     RETURN( nwtdb );                                               03495
     END.                                                           03496
                                                                    03497
(insitree) %***%                                                    02617
     %Given the stid of the origin of an inferior tree, this routine
     conects it to the data block passed as the second argument.
     Assumes other fields. have been set up already.  Cf. TRECOP and
     CREIT.%                                                        02618
     %-------------%                                               02619
     PROCEDURE(stid, nwtdb);                                        02620
     stosdb( stid, nwtdb.stpsdb);                                   02622
     stoitree( nwtdb, stid.stpsid);                                 02623
     RETURN;                                                        02624
     END.
                                                                    02625
(freintree) %***% PROCEDURE (sdbit);                                02181
     % free inferior tree:  release all structure blocks and their
     associated data blocks in the inferior tree the origin of which
     is sdbit %                                                     02182
     % this ought to work; but should the correspondence table have
     endfil?  (remgrp is not necessary.  this has been done already
     in freprop) %                                                  02183
        delgrp( sdbit, sdbit, endfil);                              02184
        % delgrp calls dsttx.  This should, in fact, call freplist
        to get rid of them all!! %                                  02185
     RETURN;                                                        02186
```

```
      END.
                                                                    02187
%structural inserts%
                                                                    01280
   (newsuc)                                                         01281
      %This routine gets a new stid and then inserts it as the suc of
      the stid passed it.  It returns the new stid.%                01282
      %-------------%                                               01283
      PROCEDURE(stid);                                              01284
      LOCAL newstd; %new stid%                                      01285
      newstd _ newrng(stid.stfile);                                 01286
      inss(stid, newstd, newstd);                                   01287
      RETURN(newstd);                                               01288
      END.
                                                                    01289
   (newsub)                                                         01290
      %This routine gets a new stid and then inserts it as the sub of
      the stid passed it.  It returns the new stid.%                01291
      %-------------%                                               01292
      PROCEDURE(stid);                                              01293
      LOCAL newstd; %new stid%                                      01294
      newstd _ newrng(stid.stfile);                                 01295
      insd(stid, newstd, newstd);                                   01296
      RETURN(newstd);                                               01297
      END.
                                                                    01298
   (insgrp)                                                         01299
      %Insert SRC1, SRC2 at TARGET according to DIR.  (SRC1 and SRC2
      are assumed to define a legal, ordered group.)%               01300
      %-------------%                                               01301
      PROCEDURE(target, dir, src1, src2);                           01302
      target _ rlevset(target, dir : dir);                          01303
      IF (target.stpsid # origin AND NOT getorf( target )) AND dir =
      levsuc THEN                                                   01304
         inss(target, src1, src2)                                   01305
      ELSE insd(target, src1, src2);                                01306
      RETURN;                                                       01307
      END.
                                                                    01308
   (inss)                                                           01309
      %Given an stid, this routine inserts a group, defined by the
      second and third arguments, as the suc of the first argument.
      First it makes the suc of STID the suc of GRP2, and updates the
      tail flag; then it makes GRP1 the suc of STID and updates the
      head and tail flags.%                                         01310
      %-------------%                                               01311
      PROCEDURE(stid, grp1, grp2);                                  01312
      IF stid.stfile # grp1.stfile THEN err($"illegal insert");     01313
      IF grp1.stfile # grp2.stfile THEN err($"illegal group");      01314
      stosuc(grp2, getsuc(stid));                                   01315
      stoftl(grp2, getftl(stid));                                   01316
      stosuc(stid, grp1);                                           01317
      stofhd(grp1, FALSE);                                          01318
      stoftl(stid, FALSE);                                          01319
      RETURN;                                                       01320
      END.
```

```
                                                                    01321
(insd)                                                              01322
   %Given an stid, this routine inserts the group defined by the
   second and third arguments down from the first argument.%        01323
   %-------------%                                                  01324
   PROCEDURE(stid, grp1, grp2);                                     01325
   LOCAL substd; %stid of sub of stid passed as argument%           01326
   IF stid.stfile # grp1.stfile THEN err($"illegal insert");        01327
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");         01328
   IF (substd _ getsub(stid)) # stid THEN                           01329
      BEGIN                                                         01330
      stofhd(substd, FALSE);                                        01331
      stoftl(grp2, FALSE);                                          01332
      END                                                           01333
   ELSE stoftl(grp2, TRUE);                                         01334
   stosuc(grp2, substd);                                            01335
   stosub(stid, grp1);                                              01336
   stofhd(grp1, TRUE);                                              01337
   RETURN;                                                          01338
   END.
                                                                    01339
(rlevset)                                                           01340
   % Determines target stid and direction for inserting
   statements.  Given an stid and a relative levadj count, returns
   an stid and levdown if to be inserted down, levsuc if to be
   inserted as successor to returned stid. %                        01341
   %---------------%                                               01342
   PROCEDURE(stid, dir);                                            01343
   LOCAL count, numb;                                               01344
   CASE dir OF                                                      01345
      =0: dir _ levsuc; %successor%                                 01346
      <0: dir _ levdown; %down%                                     01347
      ENDCASE %up number of levels indicated by dir%                01348
         WHILE dir > 0 DO                                           01349
            BEGIN                                                   01350
            stid _ getup(stid);                                     01351
            dir _ dir - 1;                                          01352
            END;                                                    01353
   RETURN(stid, dir);                                               01354
   END.
                                                                    01355
(levset)                                                            01356
   % Determines target stid and direction for inserting
   statements.  Given an stid and the address of a string
   containing u's and d's (a levadj string), returns an stid and
   -1 if to be inserted down, 0 if to be inserted as successor to
   returned stid. %                                                 01357
   %---------------%                                               01358
   PROCEDURE(stid, levstg);                                         01359
   LOCAL dir, count, numb;                                          01360
   REF levstg;                                                      01361
   dir _ 0;                                                         01362
   IF levstg.L # empty THEN                                         01363
      BEGIN                                                         01364
      count _ 1;                                                    01365
      DO                                                            01366
```

```
            CASE *levstg*[count] OF                                    01367
               ='U, ='u: BUMP DOWN dir;                                01368
               ='D, ='d: BUMP dir;                                     01369
               ENDCASE NULL                                            01370
            UNTIL (count _ count + 1) > levstg.L;                      01371
            END;                                                       01372
         CASE dir OF                                                   01373
            =0: BUMP dir; %successor%                                  01374
            >0: dir _ 0 %down%                                         01375
            ENDCASE %up number of levels indicated by dir%             01376
               WHILE (dir _ dir + 1) <= 0                              01377
                  DO stid _ getup(stid);                               01378
                                                                       01379
         RETURN(stid, dir - 1);                                        01380
         END.                                                          01381

% locate and link property %                                          03033
   (locprop) % xxx  locate destination of new property %
   PROCEDURE (destid, proptyp);                                        03035
      % locates the place after which a new property is to be
      inserted.  Returns four items: first is FALSE if error, page
      number in core if success; second is address of block in core
      (which must be frozen if you want to do anything with it!),
      third is stid or stdb of ring or property block, fourth is
      flag: TRUE if block is ring, false if DB %                       03036
      %---------------%                                               03037
      LOCAL                                                            03038
         destblk,                                                      03039
         desel,                                                        03040
         dest,                                                         03041
         destrng,                                                      03042
         stdb,                                                         03043
         ptabin,                                                       03044
         blknum,                                                       03045
         db;                                                           03046
      REF db, desel;                                                   03047
      IF NOT (destblk _ lodent(destid, rngtyp : &desel)) THEN          03048
         RETURN( FALSE, 0, 0, 0);                                      03049
      IF NOT (stdb _ desel.rsdb) THEN                                  03050
         RETURN( destblk, &desel, destid, TRUE);                       03051
      stdb.stfile _ destid.stfile;                                     03052
      dest _ destid;                                                   03053
      destrng _ TRUE;                                                  03054
      IF NOT (ptabin _ getptab(proptyp)) THEN                          03055
         RETURN( FALSE, 0, 0, 0); % illegal property type %            03056
      LOOP                                                             03057
         BEGIN                                                         03058
         IF NOT (blknum _ lodent(stdb, sdbtyp : &db)) THEN             03059
            RETURN( FALSE, 0, 0, 0);                                   03060
         CASE getptab( db.sptype ) OF                                  03061
            <= 0: RETURN( FALSE, 0, 0, 0); % illegal property type %
                                                                       03062
            < ptabin:                                                  03063
               BEGIN                                                   03064
               destblk _ blknum;                                       03065
               &desel _ &db;                                           03066
```

```
            dest _ stdb;                                              03067
            destrng _ FALSE;                                          03068
            IF (stdb.stpsdb _ db.spsdb) = 0 THEN                      03069
               REPEAT CASE(ptabin+1); % Force return %                03070
            % Repeat loop %                                           03071
            END;                                                      03072
         = ptabin: % Property already exists %                        03073
            RETURN( FALSE, 0, 0, 0);                                  03074
         ENDCASE % > ptabin %                                         03075
            RETURN( destblk, &desel, dest, destrng );                 03076
      END;                                                            03077
   END.                                                               03078
                                                                      03079
(getptab) % xxx get property table index %
 PROCEDURE (proptyp);                                                 03080
   % Given a property type, this procedure returns its index in
   the property table.  This is the number beyond which we need
   not search. If illegal proptyp (not in table) return FALSE. %
                                                                      03081
   LOCAL count;                                                       03082
   FOR count _ 1 UP UNTIL > proptab DO                                03083
      IF proptab[count] = proptyp THEN RETURN(count);                 03084
   IF proptyp IN [40000B, 77777B] THEN RETURN(proptyp);               03893
      %these types reserved for users%                                03894
   RETURN(FALSE);                                                     03085
   END.
                                                                      03086
(lnkprop)  % xxx   link a property into list structure %
PROCEDURE( destrng, % TRUE if dest a ring; FALSE if DB %              03121
   desel, % address of destination RING or DB; frozen %               03122
   dest, % stid or stdb of destination %                              03123
   sdbnew, % address of new DB; frozen %                              03124
   nwtdb % stdb of new block% );                                      03125
   REF desel, sdbnew;                                                 03126
   % Link the new block to the destination block %                    03127
      % Fill in the SPSDB or RSDB field of the dest block with the
      STPSDB of the new one %                                         03128
         IF destrng THEN                                              03129
            BEGIN                                                     03130
            sdbnew.spsdb _ desel.rsdb := nwtdb.stpsdb;                03131
            sdbnew.spsid _ dest.stpsid;                               03132
            END                                                       03133
         ELSE                                                         03134
            BEGIN                                                     03135
            sdbnew.spsdb _ desel.spsdb := nwtdb.stpsdb;               03136
            sdbnew.spsid _ desel.spsid;                               03137
            END;                                                      03138
   RETURN;                                                            03139
   END.
                                                                      03140
(unlnkprop)  % xxx   unlink a property from list structure %
PROCEDURE( sdbblk, % address of DB to be unlinked; assumed frozen
in core %                                                             03193
   stdb % of DB to be unlinked% );                                    03194
   LOCAL                                                              03195
      stid,       %of node; one of its properties is being freed.
```

```
      %                                                          03196
      sdbprop,  % value of next property field. %                03197
      rnl,        %address of ring block coresponding to stid. %
                                                                 03198
      svstdb, %STDB of a property in the node %                  03199
      fileno, % file in which block lives %                      03625
      nxtblk; % sdb corresponding to svstdb %                    03200
   REF sdbblk, rnl, nxtblk;                                      03204
   IF stdb.stpsdb = 0 THEN RETURN( FALSE);                       03205
   fileno _ stdb.stfile;                                         03626
   % link up other properties in this node around the deleted one.
   %                                                             03221
      stid _ sdbblk.spsid;                                       03222
      stid.stfile _ fileno;                                      03223
      sdbprop _ sdbblk.spsdb;                                    03224
      lodent(stid, rngtyp : &rnl);                               03225
      % Check if text block is being freed; if so, 0 name fields
      in ring %                                                  03226
         IF sdbblk.sptype = txttyp THEN                          03227
            BEGIN                                                03228
            rnl.rnamef _ rnl.rnameh _ 0;                         03229
            END;                                                 03230
      IF (svstdb _ rnl.rsdb) # stdb.stpsdb THEN                  03231
         BEGIN                                                   03232
         LOOP % over properties in this node %                   03233
            BEGIN                                                03234
            svstdb.stfile _ fileno;                              03235
            lodent(svstdb, sdbtyp : &nxtblk);                    03236
            IF (svstdb _ nxtblk.spsdb) = stdb.stpsdb THEN        03237
               BEGIN                                             03238
               nxtblk.spsdb _ sdbprop;                           03239
               EXIT LOOP;                                        03240
               END;                                              03241
            IF svstdb = 0 THEN EXIT LOOP;                        03242
            END;                                                 03243
         END                                                     03244
      ELSE                                                       03245
         BEGIN                                                   03246
         rnl.rsdb _ sdbprop;                                     03247
         END;                                                    03248
   RETURN( TRUE );                                               03249
   END.
                                                                 03250
(copnam)  % xxx copy name fields from existing to new property %
PROCEDURE( destrng, % TRUE if dest a ring; FALSE if DB %         03141
   sdbold, % address of source DB; assumed frozen in core %      03142
   oldstid, % source stid %                                      03143
   desel, % address of destination RING or DB; frozen %          03144
   dest % stid or stdb of destination % );                       03145
   LOCAL                                                         03146
      newstid, % stid of node to which new block is attached %
                                                                 03147
      oldnam, % name hash of source stid %                       03148
      oldrng,                                                    03149
      newring;                                                   03150
   REF sdbold, desel, oldrng, newring;                           03154
```

```
      % set name hash if necessary %                                    03170
        IF sdbold.sptype = txttyp THEN                                  03171
           BEGIN                                                        03172
           % Must copy name hash to ring %                              03173
           lodent( oldstid, rngtyp : &oldrng);                          03174
           oldnam _ oldrng.rnameh;                                      03175
           IF destrng THEN                                              03176
              BEGIN                                                     03177
              desel.rnamef _ oldnam # 0;                                03178
              desel.rnameh _ oldnam;                                    03179
              END                                                       03180
           ELSE                                                         03181
              BEGIN                                                     03182
              % Must load ring; text not necessarily first %            03183
              newstid _ desel.spsid;                                    03184
              newstid.stfile _ dest.stfile;                             03185
              lodent(newstid, rngtyp : &newring);                       03186
              newring.rnamef _ oldnam # 0;                              03187
              newring.rnameh _ oldnam;                                  03188
              END;                                                      03189
           END;                                                         03190
      RETURN;                                                           03191
      END.

                                                                        03192
% copy tree %                                                          03034
  (trecop) % xxx ***% %Copies a property llst.  Arguments are           03087
      (1) STID (includes STFILE field -- source file) of inferior
      tree to be copied,                                                03088
      (2) STDB (includes STFILE field -- destination file) of new
      block to which it is to be attached.  Returns stid of origin of
      inferior tree if OK; FALSE if not.                                03089
      %

                                                                        03090
      PROCEDURE (itstid, nwtdb);                                        03091
      LOCAL                                                             03092
         oldsid,                                                        03093
         newsid,                                                        03094
         ring;                                                          03682
      REF ring;                                                         03683
      % what happens if we are doing copy filtered?  if a node passes
      the filter, do we copy everything in the inferior tree
      associated with that node?  what to do? %                         03095
      ON SIGNAL ELSE RETURN(FALSE);                                     03648
      IF NOT getorf( itstid ) THEN RETURN(FALSE);                       03642
      oldsid _ itstid;                                                  03096
      newsid _ newrng(nwtdb.stfile : &ring);                            03097
      % Initialize inferior tree origin fields %                       03684
         ring.rsuc _ ring.rsub _ newsid.stpsid;                         03685
         ring.rhf _ ring.rtf _ ring.rtorgin _ TRUE;                     03686
         % rnameh and rnamef and rsid have been set in newrng.  Other
         fields were also zeroed. %                                     03687
      LOOP                                                              03098
         BEGIN                                                          03099
         %get stid's, this branch%                                      03100
         WHILE (oldsid := getsub(oldsid)) # oldsid DO                   03101
            newsid _ newsub(newsid);                                    03102
```

```
        %now copy sdb's and point to next branch%                          03103
        LOOP                                                               03104
           BEGIN                                                           03105
           copplist(oldsid, newsid);                                       03106
           upctbl(oldsid, newsid, oldsid);                                 03107
           IF oldsid = itstid THEN %tree copy complete%                    03108
              BEGIN                                                        03109
              % link up the inferior tree to the property data block
              %                                                            03110
              insitree( newsid, nwtdb);                                    03111
              RETURN( newsid );                                            03112
              END;                                                         03113
           IF NOT getftl(oldsid := getsuc(oldsid)) THEN EXIT;              03114
              %still more in plex%                                         03115
           newsid _ getsuc(newsid);                                        03116
           END;                                                            03117
        newsid _ newsuc(newsid);                                           03118
        END;                                                               03119
     END.
                                                                           03120
%Ring Utility Routines%
                                                                           0175
   (newrng) %find room for a new ring element and allocate it.
   Called with file number of file where want the new element.
   Returns                                                                 0208
      (1) STID and                                                         0209
      (2) the address of the new element.%                                 0210
      PROCEDURE (fileno);                                                  0211
      LOCAL                                                                0212
         rngblk, %counter for ring blocks%                                 0213
         rngtry, %possible block for use next%                             0214
         pgindx, %page number of where loaded%                             0215
         blkad,  %address of block where will allocate%                    0216
         fileh,  %location of file header%                                 0217
         nringl, %number of ring elements in block%                        0218
         freep,  %free list pointer in block%                              0219
         rngsta, %address of RNGST for the file%                           0220
         rngloc, %address of RNGL for the file%                            0221
         stid,   %new STID%                                                0222
         rn;     %pointer into the RNGST for the file%                     0223
      REF rn;                                                              0225
      rngsta _ $rngst + fileh _ fileheadEfileno] - $filhed;                0226
      rngloc _ fileh + $rngl;                                              0227
      %first check the ring block from which last allocated                0228
      or freed an element (number saved in RNGLST)%                        0229
      &rn _ rngsta + rnglst;                                               0230
      IF rnglst IN E0,rngm) AND                                            0231
         rn.rfexis AND rn.rfcore AND rn.rffree THEN                        0232
         BEGIN                                                             0233
         stid _                                                            0234
            nwrngb(fileh, rnglst, rn.rfcore, fileno : rngloc);             0235
         RETURN (stid, rngloc);                                            0236
         END;                                                              0237
      %else search for a block%                                            0238
      rngblk _ 0;                                                          0239
      rngtry _ -1;                                                         0240
```

```
&rn _ rngsta;                                                    0241
DO BEGIN                                                         0242
   IF rn.rfexis AND rn.rffree THEN                               0243
      IF rn.rfcore THEN                                          0244
         BEGIN                                                   0245
         stid _                                                  0246
            nwrngb(fileh, rngblk, rn.rfcore, fileno :            0247
            rngloc);                                             0248
         RETURN (stid, rngloc);                                  0249
         END                                                     0250
      ELSE rngtry _ rngblk;                                      0251
   BUMP &rn;                                                     0252
   END                                                           0253
UNTIL (rngblk _ rngblk+1) > [rngloc];                            0254
IF rngtry >= 0 THEN %load one that has room%                     0255
   BEGIN                                                         0256
   stid _ 0;                                                     0257
   stid.stfile _ fileno;                                         0258
   stid.stblk _ rngtry;                                          0259
   pgindx _ lodent( stid, rngtyp : blkad);                       0260
   stid _ nwrngb(fileh, rngtry, pgindx, fileno : rngloc);        0261
   RETURN (stid, rngloc);                                        0262
   END;                                                          0263
%must allocate a new block%                                      0264
rngblk _ 0;                                                      0265
&rn _ rngsta;                                                    0266
DO BEGIN                                                         0267
   IF NOT rn.rfexis THEN %will initialize this block%            0268
      BEGIN                                                      0269
      % Make PC if necessary; calls err if error, resets if
      catastrophe-- done here because we will write on the file
      header which does not cause a wrtpsi %                  03601
         crepc( fileno );                                     03602
      pgindx _ lodrfb(rngbas+rngblk, niltyp, fileno);            0280
      blkad _ crpgad[pgindx];                                    0281
      %finish initialization of the header%                      0282
      [blkad].fbtype _ rngtyp;  [blkad].fbind _ rngblk;          0283
      %now set up the status table%                              0284
      rn.rfexis _ TRUE;                                          0285
      rn.rfused _ rn.rffree _ fbhdl;                             0286
      rn.rfcore _ pgindx;                                        0287
      %update RNGL for the file%                                 0288
      IF rngblk > [rngloc] THEN [rngloc] _ rngblk;               0289
      %finish the initialization of the block%                   0290
      %make a free list%                                         0291
      %calculate the number of elements that will fit            0292
      in a page%                                                 0293
      nringl _ (blksiz-fbhdl) / ringl;                           0294
      freep _ blkad + fbhdl;                                     0295
      DO BEGIN                                                   0296
         [freep] _ freep - blkad + ringl;                        0297
         freep _ freep + ringl                                   0298
         END                                                     0299
      UNTIL (nringl _ nringl-1) = 1;                             0300
      %zero the last one%                                        0301
      [freep] _ 0;                                               0302
```

```
            stid _ nwrngb(fileh, rngblk, pgindx, fileno :          0303
            rngloc);                                               0304
            RETURN ( stid, rngloc);                                0305
            END;                                                   0306
         BUMP &rn;                                                 0307
         END                                                       0308
      UNTIL (rngblk _ rngblk+1) = rngm;                            0309
      %have exhausted the space available for structure%           0310
      err($"structure full") END.
                                                                   0311
   (nwrngb) %Used by newrng to actually allocate the ring element.
   Generates new SID and stores it in the ring element.  Called with
                                                                   0312
      (1) address of file header - $filhed,                        0313
      (2) the ring block number,                                   0314
      (3) the page index of the ring block,                        0315
      (4) the file number.                                         0316
      It allocates a new ring element from that block and returns
                                                                   0317
      (1) the new statement identifier (STID) and                  0318
      (2) the address of the new element.%                         0319
      PROCEDURE (fileh, rngblk, pgindx, fileno);                   0320
      LOCAL                                                        0321
         rn,     %address of ring entry in header%                 0322
         freep,  %free list pointer%                               0323
         nwrn,   %pointer to new ring block%                       0324
         nwrne,  %pointer to end of new ring block%                0325
         stid;   %stid for the new element%                        0326
      REF rn;                                                      0328
      %record the block number from which allocating the           0329
      element%                                                     0330
      rnglst _ rngblk;                                             0331
      &rn _ fileh + $rngst + rngblk;                               0332
      %check the free pointer for legality%                        0333
      IF (freep _ rn.rffree)                                       0334
         NOT IN [fbhdl,blksiz) THEN badfil(fileno);                0335
      % Make PC if necessary; calls err if error, resets if
      catastrophe-- done here because we will write on the file
      header which does not cause a wrtpsi %                      03619
         crepc( fileno );                                         03620
      freep _ freep + crpgad[pgindx]; %actual address%             0346
      rn.rffree _ [freep]; %new free pointer%                      0347
      rn.rfused _ rn.rfused + ringl; %increase used word count%    0348
      %zero the new ring element%                                  0349
      nwrne _ (nwrn _ freep) + ringl;                              0350
      DO [nwrn] _ 0 UNTIL (nwrn _ nwrn+1) = nwrne;                 0351
      % get new SID %                                             0352
      [freep].rsid _ [fileh + $sidcnt] _ [fileh + $sidcnt]         0353
      + 1;                                                         0354
      %return STID for the new element%                            0355
      stid _ 0;                                                    0356
      stid.stfile _ fileno;                                        0357
      stid.stblk _ rngblk;                                         0358
      stid.stwc _ freep-crpgad[pgindx];                            0359
      [freep].rsub _ stid;                                         0360
      RETURN (stid, freep);                                        0361
```

```
   END.
                                                                    0362
(frerng)  %free the ring element for STID given as argument%
                                                                    0363
   PROCEDURE (stid);                                                0364
   LOCAL                                                            0365
      rngloc, %location of the ring element%                        0366
      blkad,  %address of file block containing the element%        0367
      pgindx, %index of ring block%                                 0368
      cnt,    %counter for clearing element%                        0369
      pnt,    %pointer for clearing element%                        0370
      rn;    %pointer to RNGST entry%                               0371
   REF rn;                                                          0373
   %zap swork if necessary%                                         0374
      IF swork = stid THEN swork _ endfil;                          0375
      %this is done for the benefit of fechc1.  ensures that when
      a FIND fails and CCPOS is reset to position before the FIND,
      it will be reset to an existing statement or to the NULL
      string.%                                                      0376
   % Make PC if necessary; calls err if error, resets if
   catastrophe-- done here because we will write on the file
   header which does not cause a wrtpsi %                           03621
      crepc( stid.stfile );                                         03622
   rnglst _ stid.stblk;                                             0387
   pgindx _ lodent( stid, rngtyp : rngloc);                         0388
   blkad _ crpgad[pgindx];                                          0389
   %clear the element%                                              0390
   cnt _ ringl;                                                     0391
   pnt _ rngloc;                                                    0392
   DO                                                               0393
      BEGIN                                                         0394
      [pnt] _ 0;                                                    0395
      BUMP pnt;                                                     0396
      END                                                           0397
   UNTIL (cnt _ cnt-1) = 0;                                         0398
   &rn _ filehead[stid.stfile] + $rngst - $filhed +                 0399
   stid.stblk;                                                      0400
   %reduce used word count%                                         0401
   rn.rfused _ rn.rfused - ringl;                                   0402
   %add to free list%                                               0403
   [rngloc] _ rn.rffree;                                            0404
   rn.rffree _ rngloc - blkad;                                      0405
   RETURN END.
                                                                    0406
(goodrng) PROCEDURE(stid);                                          0196
   %Returns TRUE iff stid points to a ring element which is in
   use%                                                             0197
   %assumes that the file number in the stid is ok%                 0198
   LOCAL rngblk, wc, rn;                                            0199
   REF rn;                                                          0200
   %split the psid into block number and word count%                0201
   wc _ stid.stwc;                                                  0202
   rngblk _ stid.stblk;                                             0203
   &rn _ filehead[stid.stfile] + $rngst - $filhed + rngblk;         0204
   %check that block number is legal%                               0205
   RETURN(rngblk IN [0,rngm) AND rn.rfexis AND getsid(stid)#0);
```

```
                                                                    0206
    END.
                                                                    0207
%SDB Utility Routines%
                                                                    0464
  (newdb) %***% %get a new data block of type BLKTYP.  Arguments are
                                                                   02212
    (1) size of new DB,                                            02213
    (2) type of the data block                                     02214
    (3) file number from which to allocate.                        02215
    Loads the new data block and performs general initialization.
    Leaves the block loaded, though unfrozen.                      02216
    Returns the STDB of the SDB, the SDB and the index of the core
    page to which it has been loaded, %
                                                                   02217
    PROCEDURE (room, blktyp, fileno);                              02218
    LOCAL                                                          02219
      choi2,  %second choice block%                                02220
      choi3,  %third choice block%                                 02221
      least,  %number of used words in choi3%                      02222
      diff,   %excess space in garbage SDB%                        02223
      free,   %free space start%                                   02224
      sdbblk, %index in DTBST%                                     02225
      dt,     %pointer into DTBST%                                 02226
      pgindx, %page index for block in core%                       02227
      dtbsta, %address of DTBST for the file%                      02228
      dtbloc, %address of DTBL for the file%                       02229
      blkad,  %address of the block%                               02230
      blknum,                                                      02231
      sdb,                                                         02232
      sdbpt,  %pointer to SDB's in the block%                      02233
      stdb;   %STDB for the new SDB%                               02234
    REF dt;                                                        02236
    % general initialization %                                     02237
      dtbsta _ $dtbst + dtbloc _ filehead[fileno] - $filhed;       02238
      dtbloc _ dtbloc + $dtbl;                                     02239
      stdb _ 0;                                                    02240
      stdb.stfile _ fileno;                                        02241
    % Make PC if necessary; calls err if error, resets if
    catastrophe-- done here because we will write on the file
    header which does not cause a wrtpsi %                         03623
      crepc( fileno );                                             03624
    % Check if room in last used block if it's in core %           02253
      &dt _ dtbsta + dblst;                                        02254
      IF dblst IN [0,dtbm) AND dt.rfexis AND                       02255
         dt.rfcore AND dt.rffree + room <= blksiz THEN             02256
           BEGIN %we won this time%                                02257
           stdb.stblk _ dblst;                                     02258
           stdb.stwc  _ dt.rffree;                                 02259
           dt.rffree _ dt.rffree + room;                           02260
           dt.rfused _ dt.rfused + room;                           02261
           % initialize data block fields %                       02262
             blknum _ initdb( stdb, room, blktyp : sdb);           02263
           RETURN (stdb, sdb, blknum);                             02264
           END;                                                    02265
    % we have to work harder.  Look at the blocks which are in core
```

```
first (in checking tables, if an out of core block has space
without garbage collection but is not in core,  note that fact
in choi2; if it has space, but must be garbage collected, note
that fact in choi3.) %                                          02266
   choi2 _ choi3 _ -1;                                          02267
   least _ blksiz;                                             02268
   sdbblk _ 0;                                                  02269
   &dt _ dtbsta;                                                02270
   DO BEGIN                                                     02271
      IF dt.rfexis THEN %block already allocated%               02272
         IF dt.rffree + room <= blksiz THEN                     02273
            %room in free space%                                02274
            IF dt.rfcore THEN %loaded already%                  02275
               BEGIN                                            02276
               stdb.stblk _ dblst _ sdbblk;                     02277
               stdb.stwc _ dt.rffree;                           02278
               dt.rffree _ dt.rffree + room;                    02279
               dt.rfused _ dt.rfused + room;                    02280
               % initialize data block fields %                02281
                  blknum _ initdb( stdb, room, blktyp : sdb);
                                                                02282
               RETURN (stdb, sdb, blknum);                      02283
               END                                              02284
            ELSE choi2 _ sdbblk                                 02285
         ELSE %check if there is room if garbage collect%       02286
            %and page not frozen%                               02287
            IF dt.rfused + room <= blksiz AND                   02288
               dt.rfused < least AND                            02289
               (dt.rfcore = 0 OR corpst[dt.rfcore].ctfroz = 0)
                                                                02290
               THEN %new best choice%                           02291
               BEGIN                                            02292
               choi3 _ sdbblk;                                  02293
               least _ dt.rfused;                               02294
               END;                                             02295
      BUMP &dt;                                                 02296
      END                                                       02297
   UNTIL (sdbblk _ sdbblk+1) > [dtbloc];                        02298
%enough free space but not in core%                             02299
   IF choi2 >= 0 THEN                                           02300
      BEGIN                                                     02301
      stdb.stblk _ dblst _ choi2;                               02302
      &dt _ dtbsta + choi2;                                     02303
      stdb.stwc _ dt.rffree;                                    02304
      dt.rffree _ dt.rffree + room;                             02305
      dt.rfused _ dt.rfused + room;                             02306
      % initialize data block fields %                         02307
         blknum _ initdb( stdb, room, blktyp : sdb);            02308
      RETURN (stdb, sdb, blknum);                               02309
      END;                                                      02310
%if choi3 then block has enough room but may have to garbage
collect%                                                        02311
   IF choi3 >= 0 THEN                                           02312
      BEGIN                                                     02313
      stdb.stblk _ dblst _ choi3;                               02314
      lodent (stdb, sdbtyp : blkad);                            02797
```

```
            &dt _ dtbsta + choi3;                                   02316
            free _ dt.rffree + blkad;                               02317
            sdbpt _ blkad + fbhdl;                                  02318
            UNTIL sdbpt >= free DO                                  02319
               %try to find a garbage sdb that is big enough%       02320
               BEGIN                                                02321
               IF [sdbpt].sgarb AND [sdbpt].slength >= room THEN
                                                                    02322
                  %will put the new SDB here%                       02323
                  BEGIN                                             02324
                  stdb.stwc _ sdbpt - blkad;                        02325
                  IF (diff _ [sdbpt].slength - room) > 0 THEN       02326
                     BEGIN %make excess look like garbage sdb%      02327
                     [sdbpt+room].sgarb _ TRUE;                     02328
                     [sdbpt+room].slength _ diff;                   02329
                        %depends on sgarb, slength being in first
                                                                    02330
                        word of header since diff may = 1%          02331
                     END;                                           02332
                  dt.rfused _ dt.rfused + room;                     02333
                  % initialize data block fields %                 02334
                     blknum _ initdb( stdb, room, blktyp : sdb);
                                                                    02335
                  RETURN (stdb, sdb, blknum);                       02336
                  END;                                              02337
               %go to the next SDB%                                 02338
               IF (sdbpt := sdbpt + [sdbpt].slength) >= sdbpt THEN
                                                                    02339
                  badfil(fileno);                                   02340
               END;                                                 02341
            %must garbage collect the block%                        02342
            gcol(choi3, fileno);                                    02343
            stdb.stwc _ dt.rffree;                                  02344
            dt.rffree _ dt.rffree + room;                           02345
            dt.rfused _ dt.rfused + room;                           02346
            % initialize data block fields %                       02347
               blknum _ initdb( stdb, room, blktyp : sdb);          02348
            RETURN (stdb, sdb, blknum);                             02349
            END;                                                    02350
         %have to allocate a block%                                 02351
            sdbblk _ 0;                                             02352
            &dt _ dtbsta;                                           02353
            DO BEGIN                                                02354
               IF NOT dt.rfexis THEN %will initialize this block%   02355
                  BEGIN                                             02356
                  pgindx _ lodrfb(sdbblk+dtbbas, niltyp, fileno);   02357
                  blkad _ crpgad[pgindx];                           02358
                  %finish initialization of block header%          02359
                  [blkad].fbind  _ sdbblk;                          02360
                  [blkad].fbtype _ sdbtyp;                          02361
                  %now set up the status table entry%               02362
                  dt.rfcore _ pgindx;                               02363
                  dt.rfexis _ TRUE;                                 02364
                  dt.rfused _ dt.rffree _ fbhdl+room;               02365
                  %update DTBL for the file%                        02366
                  IF sdbblk > [dtbloc] THEN [dtbloc] _ sdbblk;      02367
```

```
            dblst _ sdbblk;                                          02368
            stdb.stblk _ dblst;                                      02369
            stdb.stwc _ fbhdl;                                       02370
            % initialize data block fields %                         02371
               blknum _ initdb( stdb, room, blktyp : sdb);           02372
            RETURN (stdb, sdb, blknum);                              02373
            END;                                                     02374
         BUMP &dt;                                                   02375
         END                                                         02376
      UNTIL (sdbblk _ sdbblk+1) = dtbm;                              02377
   %have exhausted data blocks%                                      02378
   err($"data storage full") END.
                                                                    02379
(initdb) %***% PROCEDURE ( stdb, sdbsiz, blktyp);                   02380
   LOCAL blknum, sdb, end, sdbpt;                                   02381
   REF sdb;                                                         02382
   % load the block into core %                                     02383
      blknum _ lodent( stdb, sdbtyp : &sdb);                        02384
   % Initialization of general fields in dbheader by newdb %        02385
      end _ (sdbpt _ &sdb) + sdbhdl;                                02386
      DO [sdbpt] _ 0 UNTIL (sdbpt _ sdbpt+1) = end;                 02387
      sdb.sgarb _ FALSE;                                            02388
      sdb.slength _ sdbsiz;                                         02389
      sdb.sinit _ cinit;                                            02390
      sdb.stime _ gtadcall();                                       02392
      sdb.sptype _ blktyp;                                          02393
   RETURN( blknum, &sdb);                                           02394
   END.
                                                                    02395
(gcol) %***% %called by newsdb to garbage collect the sdb block
whose block number and file number are passed as arguments.%
                                                                    02428
   PROCEDURE (sdbblk, fileno);                                      02429
   LOCAL                                                            02430
      pgindx,  %page index for core page holding the block%         02431
      blkad,  %address of the block%                                02432
      blkfre, %address of free space in block%                      02433
      freewd, %pointer to first garbage block%                      02434
      sdbsiz, %size of SDB%                                         02435
      pt,     %pointer to SDB%                                      02436
      stdb,   %stdb used in fixing up linkages%                     02437
      tstsdb,                                                       02438
      savsdb,                                                       02439
      elem,                                                         02440
      stid,   %stid for fix up's%                                   02441
      dt;    %pointer to DTBST entry for the block%                 02442
   REF dt, elem;                                                    02443
   % Initialize work stids and stdbs with file number %             03697
      stdb _ tstsdb _ savsdb _ stid _ 0;                            02445
      stdb.stfile _ tstsdb.stfile _  savsdb.stfile _ stid.stfile _
      fileno;                                                       02446
      stdb.stblk _ tstsdb.stblk _ savsdb.stblk _ sdbblk;            02448
   % get the data block status table entry location. %             03698
      &dt _ filehead[fileno] + $dtbst + sdbblk - $filhed;           02449
   % load the block into core and freeze it %                       03699
      pgindx _ lodent( stdb, sdbtyp : blkad);                       02450
```

```
   frzblk(pgindx, 1);                                              02451
ON SIGNAL ELSE frzblk(pgindx, -1); %release frozen page%           02452
blkfre _ blkad + dt.rffree; %start of free space%                  02453
freewd _ blkad + fbhdl; %first SDB%                                02454
%move to first garbage sdb%                                        02455
UNTIL [freewd].sgarb OR freewd >= blkfre DO                        02456
   freewd _ freewd + [freewd].slength;                             02457
% In the following loop, pt will point to the first word of the
non-garbage block to be moved and freewd will point to the
first free space location which will be filled up. %               03700
pt _ freewd;                                                       02458
WHILE pt < blkfre % the first word in free space for block % DO
                                                                   02459
   BEGIN                                                           02460
   IF (sdbsiz _ [pt].slength) = 0 THEN badfil(fileno);             02461
   IF NOT [pt].sgarb THEN                                          02462
      BEGIN                                                        02463
      mvbfbf(pt, freewd, sdbsiz); %move up the next good           02464
      sdb%                                                         02465
      % ******* %                                                  03701
      % This code saves the old internal pointer-name (stpsdb)
      of the property block which has just been moved up in
      tstsdb.stpsdb.  It then looks at the places which may
      have pointed to it and replaces the pointer with its new
      pointer name (stdb.stpsdb).  If the block moved was the
      first in a property list, the ring's rsdb field will be
      its old name and will be changed;  if not, we must thread
      through the property list, loading the property blocks
      until we come to the one which points to the one just
      moved.  We change the pointer and exit the loop.  (If we
      reach the end of the list we have a bad file: nobody
      pointed to the block!)  One other place could point to a
      property block:  that is the block's inferior tree.  If
      the block has one we change its pointer as well. %           03702
      %store new stdb -- correct property chain%                   02466
         stid.stpsid _ [freewd].spsid;                             02467
         stdb.stwc _ freewd-blkad; % the new stdb %                02468
         tstsdb.stwc _ pt - blkad; % the old stdb %                02469
         lodent( stid, rngtyp : &elem);                            02470
         IF elem.rsid = 0 THEN err($"Bad statement
         identifier");                                             02471
         IF tstsdb.stpsdb = (savsdb.stpsdb _ elem.rsdb) THEN
                                                                   02472
            elem.rsdb _ stdb.stpsdb                                02473
         ELSE                                                      02474
            LOOP % this is in the middle of a property list %
                                                                   02475
               BEGIN                                               02476
               lodent( savsdb, sdbtyp : &elem);                    02477
               IF (tstsdb.stpsdb = (savsdb.stpsdb_elem.spsdb))
               THEN                                                02478
                  BEGIN                                            02479
                  elem.spsdb _ stdb.stpsdb;                        02480
                  EXIT LOOP;                                       02481
                  END                                              02482
               ELSE                                                02483
```

```
                        IF savsdb.stpsdb = 0 THEN badfil(fileno);
                                                                    02484
                     END;                                           02485
               % Check for inferior tree -- relink if present %     03688
                  IF (stid.stpsid _ [freewd].sitpsid) THEN          03689
                     BEGIN                                          03690
                     lodent(stid, rngtyp : &elem);                  03691
                     IF elem.rsid=0 THEN                            03692
                        err($"Bad statement identifier");           03696
                     IF elem.rtorgin THEN                           03693
                        elem.rsdb _ stdb.stpsdb;                    03694
                     END;                                           03695
               % ******* %                                         03703
               freewd _ freewd + sdbsiz;                            02486
               END;                                                 02487
            pt _ pt + sdbsiz;                                       02488
            END;                                                    02489
         %tinally update the status table%                          02490
         dt.rffree _ freewd - blkad;                                02491
         IF dt.rffree # dt.rfused THEN badfil(fileno);              02492
         frzblk(pgindx, -1);                                        02493
         RETURN END.
                                                                    02494
% Miscellaneous support procedures %                                02811
  % Freeze blocks in core %                                        02812
     (frzrfb) %This routine is called for all freezing and thawing of
     random file blocks.  Arguments are                             0134
        (1) file number,                                            0135
        (2) block number in the file,                               0136
        (3) a 1 to freeze the block, and a -1 to thaw it.           0137
        RERROR is called if that block is not in core.  Anyone      0138
        who freezes a block must be sure it is thawed -- and        0139
        only once.%
                                                                    0140
        PROCEDURE (fileno, blkn, fr);                               0141
        LOCAL                                                       0142
           pgindex, %file block page counter%                       0143
           ct;  %pointer to CORPST entry%                           0144
        REF ct;                                                     0145
        pgindex _ 1;                                                0146
        &ct _ $corpst + 1;                                          0147
        DO                                                          0148
           BEGIN                                                    0149
           IF ct.ctfull AND                                         0150
              ct.ctfile = fileno AND                                0151
              ct.ctpnum = blkn THEN                                 0152
                 BEGIN                                              0153
                 IF ct.ctfroz + fr NOT IN [0,7] THEN                0154
                    err($"Block frozen too many times in FRZRFB");
                                                                    01500
                 ct.ctfroz _ ct.ctfroz + fr;                        0155
                 RETURN;                                            0156
                 END;                                               0157
           BUMP &ct;                                                0158
           END                                                      0159
        UNTIL (pgindex _ pgindex+1) > rfpmax;                       0160
```

```
      err($"Block not found in FRZRFB");                                0161
      END.                                                              0162
                                                                        0163
   (frzblk) %Freeze block given as arguments                            0164
      (1) the index in CRPGAD of the block, and                         0165
      (2) the 1 or -1 for freeze or thaw.%                              0166
                                                                        0167
      PROCEDURE (pgindx, fr);                                           0168
      LOCAL ct; REF ct;                                                 0169
      &ct _ $corpst + pgindx;                                           0170
      IF ct.ctfroz + fr NOT IN [0,7] THEN                               0171
         err($"Block frozen too many times in frzblk");                01501
      ct.ctfroz _ ct.ctfroz + fr;                                       0172
      RETURN END.                                                       0173
                                                                        0174
% Bad file %                                                           02813
   (badfil) %called when find something screwed up in the file%         0129
      PROCEDURE (fileno);                                               0130
      bfilno _ fileno; %save it for SIGNAL%                             0131
      SIGNAL(-5, $"Bad File");                                          0132
      END.                                                              0133
%correspondence table manipulation%
                                                                        01459
   (upctbl)                                                             01460
      %Given three stid's, this routine will update occurrences of
      the first stid, using rplstid as the stid that contains the
      text corresponding to oldsid, and newsid as the stid that is
      the replacement for oldsid.  (rplstid should be endfil if the
      original text has been eliminated.)%                              01461
      %-------------%                                                   01462
      PROCEDURE(oldsid, rplstid, newsid);                               01463
      LOCAL list, listnd;                                               01464
      REF list;                                                         01465
      IF clstad = 0 OR [clstad].clbuff = 0 THEN RETURN;                 01466
      &list _ [clstad].clbuff;                                          01467
      listnd _ &list + [clstad].clcnt * cll;                            01468
      FOR &list UP cll UNTIL >= listnd                                  01469
      DO                                                                01470
         IF list.clst1 = oldsid THEN                                    01471
            BEGIN                                                       01472
            list.clst2 _ newsid;                                        01473
            list.clst1 _ rplstid;                                       01474
            END                                                         01475
         ELSE                                                           01476
            IF list.clst2 = oldsid THEN                                 01477
               list.clst2 _ newsid;                                     01478
      RETURN;                                                           01479
      END.
                                                                        01480
%File header location%
                                                                        01125
   (filhdr) PROCEDURE (fileno);                                         01126
      %returns the address of the file header for the file whose
      number is passed%                                                 01127
      %-------------%                                                   01128
      LOCAL fl;                                                         01129
```

```
      REF fl;                                                          01130
      &fl _ (fileno-1)*filstl + $filst;                                01131
      RETURN (crpgad[fl.flhead] + fbhdl) END.
                                                                       01132
%test and set bounds of structure%
                                                                       01779
   (grptst)                                                            01780
      %Given two stid's, this routine checks that they specify a
      legal group; it also returns them ordered (GRP1,GRP2). If the
      stid's do not form a legal group, an err($"illegal group") is
      issued.%                                                         01782
      %-------------%                                                  01783
      PROCEDURE(stid1, stid2);                                         01784
      LOCAL t1, %working stid1%                                        01785
         t2; %working stid2%                                           01786
      IF (stid1.stpsid # stid2.stpsid AND (stid1.stpsid = orgstid OR
      stid2.stpsid = orgstid))                                         01787
      OR stid1.stfile # stid2.stfile THEN err($"illegal group");
                                                                       01788
      t1 _ stid1; t2 _ stid2;                                          01789
      LOOP                                                             01790
         BEGIN %is stid2 on same level, and after, stid1%              01791
         IF t1 = stid2 THEN RETURN(stid1, stid2);                      01792
         IF getftl(t1) THEN LOOP                                       01793
            BEGIN %is stid1 on same level after stid2%                 01794
            IF t2 = stid1 THEN RETURN(stid2, stid1);                   01795
            IF getftl(t2) THEN err($"invalid group selection");
                                                                       01796
            t2 _ getsuc(t2);                                           01797
            END;                                                       01798
         t1 _ getsuc(t1);                                              01799
         END;                                                          01800
      END.
                                                                       01801
% write Pseudo interrupt for file pages%
                                                                        0896
   % write psi %                                                       02814
      (wrtpsi) PROCEDURE;                                               0897
         % save the accumulators %                                     0898
         svac1 _ r1; r1 _ $svacs; !BLT r1, svacse;                     0899
         s _ s + 40000040B;                                            0900
         wrpi();                                                       0901
         !HRLZI r1, svacs;                                             0902
         !BLT r1, 17B;                                                 0903
         r1 _ svac1;                                                   0904
         !JSYS debrk;                                                  0905
         END.
                                                                       0906
      (wrpi) % write pseudo interrupt routine %                        03920
         PROCEDURE;                                                    03921
         LOCAL trpw, trpd, tpga, tpgX, fl, ct, rf, topc, flg, drastic,
         lvtadd;                                                       03922
         REF fl, ct, rf, lvtadd;                                       03923
         drastic _ FALSE;                                              03924
         &lvtadd _ levtab[((chntab[17].LH) - 1)];                      03925
         % read the trap words %                                       03926
```

```
r1 _ 4B5; !JSYS gtrpw;                                          03927
trpw _ r1; trpd _ r2;                                           03928
% make sure that really is a write trap %                       03929
IF NOT trpw .A 4B6 THEN !JSYS haltf;                            03930
tpga _ trpw .A 777B3;                                           03931
FOR tpgx _ 1 UP UNTIL > rfpmax DO                               03932
   IF crpgad[tpgx] = tpga THEN                                  03933
      BEGIN                                                     03934
      % set up pointers %                                       03935
      &ct _ $corpst + tpgx;                                     03936
      &fl _ (ct.ctfile-1)*filstl + $filst;                      03937
      &rf _ crpgad[fl.flhead] + fbhdl - $filhed;                03938
      &rf _ &rf + $rfbs + ct.ctpnum;                            03939
      IF rf.rfpart THEN % page from partial copy %              03940
         BEGIN                                                  03941
         IF fl.flpcread THEN                                    03942
            BEGIN                                               03943
            dismes(2, $"Cannot write on this file");            03944
            lvtadd _ $wpiabt;                                   03945
            RETURN;                                             03946
            END;                                                03947
         topc _ FALSE;                                          03948
         r2 _ 14B10;                                            03949
         END                                                    03950
      ELSE                                                      03951
         BEGIN % page from original file %                      03952
         % check if have a partial copy %                       03953
         drastic _ FALSE;                                       03954
         IF NOT fl.flpart THEN % dont have one %                03955
            IF (NOT fl.flbrws AND NOT lkfile(&fl)) OR NOT
            makepc(&fl, ct.ctfile, FALSE, $lit : drastic) THEN
                                                                03956
               BEGIN                                            03957
               IF NOT drastic AND NOT fl.flbrws THEN lkun(&fl,
               $lit); %change user setable word to unlock%
                                                                03958
               dismes(2, $lit);                                 03959
               lvtadd _ IF drastic                              03960
                  THEN $supervisor                              03961
                  ELSE $wpiabt;                                 03962
               RETURN END;                                      03963
         IF fl.flpcread THEN                                    03964
            BEGIN                                               03965
            dismes(2, $"Cannot write on this file");            03966
            lvtadd _ $wpiabt;                                   03967
            RETURN;                                             03968
            END;                                                03969
         topc _ TRUE;                                           03970
         rf.rfpart _ TRUE;                                      03971
         r2 _ 1004B8;                                           03972
         END;                                                   03973
      % change access %                                         03974
      r1.LH _ 4B5;                                              03975
      r1.RH _ tpga / 512;                                       03976
      !spacs(r1);  % 60B %                                      03977
      %must check if rm,w  or write%                            03978
```

```
            IF trpw .A 10B6 THEN  %read modify write%                    03979
               BEGIN                                                     03980
               [trpw.RH] _ [trpw.RH];   %must touch page to make
               private%                                                  03981
               END                                                       03982
            ELSE [trpw.RH] _ trpd;  %do write for write only case!%
                                                                         03983
            % map to pc if needed %                                      03984
            IF topc THEN                                                 03985
               BEGIN                                                     03986
               r1.RH _ tpga/512;                                         03987
               r1.LH _ 4B5;                                              03988
               r2.RH _ ct.ctpnum;                                        03989
               r2.LH _ fl.flpart;                                        03990
               !pmap(r1, r2, 14B10);                                     03991
               IF tops20flag THEN                                        03992
                  %tops20 releases the page from the map, get it
                  back%                                                  03993
                  BEGIN                                                  03994
                  !EXCH r1,r2;                                           03995
                  !pmap();                                               03996
                  END;                                                   03997
               END;                                                      03998
            RETURN;                                                      03999
            END;                                                         04000
         % the write is not into a file page %                           04001
         r1 _ $"Illegal write at location ";                             04002
         !HRLI r1,7B2;                                                   04003
         !JSYS psout;                                                    04004
         r1 _ 101B; %primary output file%                                04005
         r2 _ (lvtadd - 1) .A 18M; % the location %                      04006
         r3 _ 8; %base 8%                                                04007
         !JSYS nout; % write the number %                                04008
         LOOP !JSYS haltf;                                               04009
         END.
                                                                         04010
      (wpiabt)PROC;                                                      01122
         GOTO STATE;                                                     01123
         END.                                                            01124
% lock procedures %                                                     02815
      (lkun) PROCEDURE (fl, astrng);                                      0973
         %Resets user setable word to unlock%                             0974
         LOCAL lokdir, lokinit;                                           0975
         REF fl, astrng;                                                  0976
         r1 _ fl.florig;                                                  0977
         r2 _ 1000024B; %read one word at 24B%                            0978
         r3 _ $r3;                                                        0979
         !JSYS gtfdb;                                                     0980
         lokinit _ r3.lkinit;                                             0982
         lokdir _ drcdecode(r3.lkdirn, fl.flpcst, lokinit);               0981
         IF logdirno = lokdir AND lokinit = cinit THEN                    0984
            IF NOT setfdb(0, 0, &fl) THEN                                 0985
               %If user setable word has not been set back here, file is
               probably bad%                                             0986
               *astrng* _ *astrng*, "File is bad";                       0987
         RETURN;                                                         0988
```

```
    END.

                                                                    0989
(lockfile) PROCEDURE(fileno);                                       0990
    IF NOT lkfile(flntadr(fileno)) THEN                             0991
        BEGIN                                                       0992
        dismes(2, $lit);                                            0993
        RETURN(FALSE);                                              0994
        END;                                                        0995
    RETURN(TRUE) END.
                                                                    0996
(lkfile) PROCEDURE(fl);                                             0997
    LOCAL                                                           0998
        dir, lkword, int, cnt, byt, bptr, bptr0, bptr1, char, mask,
        value;                                                     01504
    LOCAL STRING intstr[5], dirname[30];                            0999
    REF fl;                                                        01000
    !gtfdb( fl.florig, 1000024B, $r3);                             01004
    lkword _ r3;                                                   01005
    IF NOT maywrt(0, &fl)
    OR NOT fl.flaccm .A accmask[lkword.acctyp] THEN                01006
        BEGIN                                                      01007
        *lit* _ "No Write Access To ", *[fl.flastr]*;              01008
        RETURN(FALSE);                                             01009
        END;                                                       01010
    IF lkword.lkdirn NOT= 0 OR                                     01011
        lkword.lkinit NOT= 0 THEN %file is locked%                 01012
        BEGIN                                                      01013
        int _ lkword.lkinit;                                       01015
        dir _ drcdecode(lkword.lkdirn, int, fl.flpcst);            04168
        %See If user has locked it..if so, let it go%              01016
        IF logdirno = dir AND cinit = int THEN RETURN(TRUE);       01018
        IF int .A 4B6 THEN %maybe locked with old style initials%
                                                                   01019
            BEGIN                                                  01020
            *intstr* _ NULL;                                       01021
            trnsint(cinit, $intstr);                               01022
            IF intstr.L = 3 AND intstr[1].oldint = int THEN        01023
                RETURN(TRUE);                                      01024
            END;                                                   01025
        *lit* _ *[fl.flastr]*;                                     01026
        lockid($lit,  lkword.lkdirn, int); %get lock ident string%
                                                                   01027
        RETURN(FALSE);                                             01028
        END;                                                       01029
    %Now get directory from PC name%                               01030
        bptr _ chbptr(0) + fl.flpcst;                              01031
        bptr0 _ bptr1 _ chbptr(0) + $dirname;                      01032
        WHILE  ^bptr # "< DO NULL;                                 01033
        WHILE (char _^bptr) # "> DO ^bptr1 _ char;                 01034
        dirname.L _ slngth(bptr0, bptr1);                          04170
        ^bptr1 _ 0;                                                01035
        IF NOT (dir _ stdircall($dirname, 0)) THEN err($"Illegal PC
        Name");                                                    04169
    mask _ 0;                                                      01046
    mask.lkinit _ mask.lkdirn _ 36M;                               01047
    value.lkinit _ cinit;                                          01048
```

```
      value.lkdirn _ dir;                                              01049
      chnfdb(fl.florig, 24B, mask, value);                             01503
      RETURN(TRUE);                                                    01051
      END.
                                                                       01502
   (lockid) PROCEDURE(astr, dircode, initials);                        01052
      %append "Being Modified By dirnumnam (init)" to astr%            01053
      REF astr;                                                        01054
      LOCAL STRING intstr[5];                                          01055
      LOCAL cnt, byt, dirnum;                                          01056
      dirnum _ drcdecode(dircode, 0, 0);                               04167
      *astr* _ *astr*, " Being Modified By ";                          01057
      IF dirnum AND SKIP !dirst(byt _ chbptr(astr.L) + &astr, dirnum)
      AND                                                              01060
         (cnt _ slngth(byt, r1)) + astr.L <= astr.M THEN               01061
            astr.L _ astr.L + cnt;                                     01062
      *intstr* _ NULL;                                                 01063
      IF initials .A 486 THEN %old style initials%                     01064
         BEGIN                                                         01065
         intstr[1].oldint _ initials;                                  01066
         intstr.L _ 3;                                                 01067
         END                                                           01068
      ELSE trnsint(initials, $intstr);                                 01069
      *astr* _ *astr*, SP, '(, *intstr*, ');                           01070
      RETURN END.
                                                                       01071
% make PC %                                                           02816
   (crepc) % create a PC if necessary%
   PROCEDURE( fileno );                                                03895
      % Used by procedures which write on the file header page to
      create PC because they do not go through the wrtpsi mechanism.
      Calls err if trouble.  If catastrophe, does a dismes and resets
      NLS. %                                                           03896
      LOCAL fl, drastic;                                               03897
      REF fl;                                                          03898
      IF NOT filepart[fileno] THEN %Make pc if neccessary%             03899
         BEGIN                                                         03900
         drastic _ FALSE;                                              03901
         &fl _ (fileno-1)*filstl + $filst;                             03902
         IF (NOT fl.flbrws AND NOT lkfile(&fl)) OR                     03903
            NOT makepc(&fl, fileno, FALSE, $lit : drastic) THEN        03904
               BEGIN                                                   03905
               IF NOT drastic AND NOT fl.flbrws THEN                   03906
                  lkun(&fl, $lit); %change user setable word to
                  unlock%                                              03907
               IF drastic THEN                                         03908
                  BEGIN                                                03909
                  % catastrophic error; reset system %                 03910
                  dismes(2, $lit);                                     03911
                  pause( 4000 );                                       03912
                  nlsrst();                                            03913
                  END;                                                 03914
               err($lit);                                              03915
               END;                                                    03916
         END;                                                          03917
      RETURN;                                                          03918
```

```
    END.

                                                                    03919
(makepc)PROC(fl, fileno, undmodflag, astr);                         04011
    REF fl, astr;                                                   04012
    LOCAL jfn, i, flg, delflag;                                     04013
    %Create a PC for te indicated file, return any errors in astr.
                                                                    04014
    REturn TRUE if ok, FALSE otherwise%                             04015
    delflag _ FALSE;  %init%                                        04016
    flg _ getgtjflg(write, chkpcf, 0);                              04017
    IF (fl.flpart _ r1 _ sgtjfn(flg, fl.flpcst, &astr)) THEN BEGIN
                                                                    04018
        r2 _ 1000001B;                                              04019
        r3 _ $r3;                                                   04020
        !JSYS gtfdb;                                                04021
        IF SKIP !TLNE r3,60000B AND NOT undmodflag THEN             04022
            BEGIN  %exists already, not deleted, and not called to
            undelete mods%                                          04023
            thwfil(fileno);                                         04024
            delfil(fileno);                                         04025
            IF nlmode = fulldisplay THEN alldsp() ELSE tda.dacsp _
            endfil;                                                 04026
            *astr* _ "File Locking Conflict--Please Reload File";
                                                                    04027
            RETURN (FALSE, TRUE);                                   04028
            END;                                                    04029
        IF SKIP !TLNN r3,40000B THEN  %deleted%                     04030
            BEGIN %must undelete file%                              04031
            !HRLI r1,1; !HRLZI r2,40000B; r3 _ 0; !JSYS chfdb;      04032
            delflag _ TRUE;                                         04033
            END;                                                    04034
        END;                                                        04035
    IF NOT fl.flpart OR                                             04036
        NOT sysopen(fl.flpart, readwrite, random, &astr) OR         04037
        NOT sysclose(fl.flpart .V 4B11, &astr) THEN                 04038
            BEGIN                                                   04039
            IF (r1 _ fl.flpart) THEN                                04040
                BEGIN %get rid of jfn%                              04041
                IF SKIP !JSYS closf THEN NULL                       04042
                ELSE                                                04043
                    %closf jsys failed (file may already be closed; set
                    up r1 for rljfn%                                04044
                    r1 _ fl.flpart;                                 04045
                IF SKIP !JSYS rljfn THEN NULL;                      04046
                END;                                                04047
            fl.flpart _ 0;                                          04048
            filepart[fileno] _ FALSE;                               04049
            RETURN(FALSE, FALSE);                                   04050
            END;                                                    04051
    filepart[fileno] _ TRUE;                                        04052
    IF NOT sysopen(fl.flpart, readwrite, random, &astr) THEN        04053
        BEGIN %Wait fo 1 sec and try it again in case of a timing
        problem%                                                    04054
        r1 _ 1000;                                                  04055
        !JSYS dismS;                                                04056
        IF NOT sysopen(fl.flpart, readwrite, random, &astr) THEN
```

```
                                                                    04057
      BEGIN                                                         04058
      IF (r1 _ fl.flpart) THEN %get rid of jfn%                     04059
         BEGIN                                                      04060
         IF SKIP !JSYS closf THEN NULL                              04061
         ELSE                                                       04062
            %closf jsys failed (file may already be closed; set
            up r1 for rljfn%                                        04063
            r1 _ fl.flpart;                                         04064
         IF SKIP !JSYS rljfn THEN NULL;                             04065
         END;                                                       04066
      fl.flpart _ 0;                                                04067
      filepart[fileno] _ FALSE;                                     04068
      RETURN(FALSE, FALSE);                                         04069
      END;                                                          04070
   END;                                                             04071
IF undmodflag THEN                                                  04072
   BEGIN  %undeleting modifications%                                04073
   IF NOT delflag THEN err($"Modifications not deleted.");
                                                                    04074
   %set the lock word%                                              04075
      lkfile(&fl);                                                  04116
   %read in the PC header%                                          04080
      IF NOT rdhdr(fileno, &astr) THEN err(&astr);                  04081
   END                                                              04082
ELSE                                                                04083
   BEGIN  %regular PC creation%                                     04084
   % change protection of pc to be same as file %                  04085
      !gtfdb(fl.florig, 1000004B, $r3); %protecton of original
      file%                                                         04086
      r1.LH _ 4B; %offset for protection word)%                     04087
      r1.RH _ fl.flpart; %jfn for partial copy%                     04088
      !chfdb(r1, 777777B, r3); %change righthalf bits only%
                                                                    04089
   %map header out to the pc%                                       04090
      r1.LH _ 4B5;                                                  04091
      r1.RH _ crpgad[fl.flhead] / 512;                              04092
      r2.LH _ fl.flpart;                                            04093
      r2.RH _ 0;                                                    04094
      r3 _ 14B10;                                                   04095
      !JSYS pmap;                                                   04096
      IF tops20flag THEN                                            04097
         BEGIN                                                      04098
         !EXCH r1,r2;                                               04099
         !pmap();                                                   04100
         END;                                                       04101
   %DELETE OLD PAGES EXCEPT FOR THE HEADER%                         04102
      jfn _ fl.flpart;                                              04103
      FOR i _ 1 UP UNTIL >= 512 DO                                  04104
         BEGIN                                                      04105
         r1.LH _ jfn;  r1.RH _ i;                                   04106
         !rpacs(r1);                                                04107
         IF r2 .A 1B10 THEN %page exists -- get rid of it%
                                                                    04108
            BEGIN                                                   04109
            r2.LH _ jfn;  r2.RH _ i;                                04110
```

```
                    !pmap(-1, r2, 0);                                   04111
                    END;                                                04112
                END;                                                    04113
            END;                                                        04114
        RETURN(TRUE, FALSE) END.
                                                                        04115
FINISH of filmnp                                                          04
```