PSI DE MI.

PSIDENT PSIDENT PSIDENT PSIDENT PSIDENT PSIDENT PSIDENT
PSIDENT PSIDENT PSIDENT PSIDENT PSIDENT PSIDENT PSIDENT

```
< NLS, PSIDENT.NLS;4, >, 9-SEP-74 16:29 CHI ;;;;( NLS, PSIDENT.NLS;4, ),
15-MAY-74 16:30 KEV ;
FILE psident % L10 <rel-nls>psident %% (L10,) (rel-nls,psident.rel,) %
                                                                        02
% Parser-support for IDENTIFICATION subsystem %                         03
(xiloaidfil)   % load Master Ident-File file %
PROCEDURE (resultptr, parsemode);                                       04
   REF resultptr;                                                       06
   LOCAL STRING infostr[1000];                                          07
   CASE parsemode OF                                                    08
      = parsing:                                                        09
         BEGIN                                                          010
         IF NOT idfno THEN                                              012
            BEGIN                                                       013
            idfno ← loaidfil();                                        0914
            idModflag ← idmodified ← FALSE;                            0821
            IF ckiwheel THEN                                            015
               BEGIN                                                    016
               ckiwheel ← FALSE;                                        017
               identwheel ← FALSE;                                      018
               IF NOT ckident($initsr, $infostr, idfno) THEN            019
                  err($"Illegal user IDENT");                           020
               IF geticapabilities($infostr, $infostr, 0,0) THEN        021
                  IF FIND SF(*infostr*) ["Ident-Wheel"] THEN            022
                     identwheel ← TRUE;                                 023
               END;                                                     024
            END;                                                       0916
         IF idcrec THEN freestring(idcrec, $dspblk);                    026
         idcrec ← getstring(2000, $dspblk);                             027
         IF idcident THEN freestring(idcident, $dspblk);                028
         idcident ← getstring(20, $dspblk);                             029
         idcrtype ← 0;                                                  030
         END;                                                           031
   ENDCASE;                                                             032
   RETURN( &resultptr );                                                033
   END.
                                                                        034
(xicloidfil)   % close Master Ident-File file %
PROCEDURE (resultptr, parsemode);                                       035
   REF resultptr;                                                       037
   CASE parsemode OF                                                    038
      = parsing:                                                        039
         BEGIN                                                          040
         IF idfno THEN                                                  042
            BEGIN                                                       043
            [flntadr(idfno)].flnoclos ← FALSE;                         0471
            close(idfno);                                               044
            idfno ← 0;                                                  045
            IF idcrec THEN freestring(idcrec := 0, $dspblk);            046
            IF idcident THEN freestring(idcident := 0, $dspblk);        047
            END;                                                        048
         END;                                                           049
      ENDCASE;                                                          050
   RETURN( &resultptr );                                                051
   END.
                                                                        052
```

```
(xiupdate)     % update modified record %
PROCEDURE (resultptr, parsemode, now);                                  053
   LOCAL STRING badids[100], workstr[100];                             0488
   REF resultptr, now;                                                  055
   CASE parsemode OF                                                    056
      = parsing:                                                        057
         BEGIN                                                          058
         IF NOT idcrec OR NOT [idcrec].L THEN err($"no record loaded!");
                                                                       0895
         IF now THEN                                                   0472
            dismes(1, $"Updating Master Ident-File.  Please wait.")     059
         ELSE                                                          0473
            dismes(1, $"Partially Updating Master Ident-File.  Please
            wait.");                                                   0474
         %check for presence of necessary fields%                       061
            %ident, name, coord, org, mail address%                    0487
               IF NOT [idcident].L THEN                                0490
                  err($"Ident is missing--update not performed");      0493
               %ident if not new record%                               0579
                  IF NOT nwrecflag THEN                                0580
                     BEGIN                                             0581
                     getiid ( idcrec, $workstr, 0,0);                  0582
                        %don't use idcident since if it is different
                        then getiid, it is new%                        0583
                     IF NOT oldid($workstr, idfno) THEN                0584
                        BEGIN                                          0585
                        *badids* ← "Bad Ident in (old) IDENT field: ",
                        *badids*, " -- Update not performed";          0586
                        err ( $badids );                               0587
                        END;                                           0588
                     END;                                              0589
               getinam ( idcrec, $workstr, 0,0);                       0497
               IF NOT workstr.L THEN                                   0498
                  err($"name must be presant--update not performed");
                                                                       0499
               getiadd ( idcrec, $workstr, 0,0);                       0500
               IF NOT workstr.L THEN                                   0501
                  err($"Address must be presant--update not performed");
                                                                       0502
               IF idcrtype =  indtyp THEN                              0506
                  BEGIN  %organization%                                0507
                  getiorg ( idcrec, $workstr, 0,0);                    0503
                  IF NOT workstr.L THEN                                0504
                     err($"Organization must be presant--update not
                     performed");                                      0505
                  IF NOT oldid($workstr, idfno) THEN                   0529
                     BEGIN                                             0530
                     *badids* ← "Bad Ident in ORGANIZATION field: ",
                     *badids*, " -- Update not performed";             0531
                     err ( $badids );                                  0532
                     END;                                              0533
                  END                                                  0506
               ELSE                                                    0509
                  BEGIN  %coordinator%                                 0510
                  geticord ( idcrec, $workstr, 0,0);                   0511
                  IF NOT workstr.L THEN                                0512
```

```
            err($"Coordinator must be presant--update not
            performed");                                          0513
         IF NOT oldid($workstr, idfno) THEN                       0546
            BEGIN                                                 0547
            *badids* ← "Bad Ident in COORDINATOR field: ",
            *badids*, " -- Update not performed";                 0548
            err ( $badids );                                      0549
            END;                                                  0550
         END:                                                     0514
   %verify the idents in the record%                              0475
      %group/org or individual ? %                                0476
         IF idcrtype = indtype THEN  %individual%                 0478
            BEGIN                                                 0515
            %groups%                                              0483
               getigrps ( idcrec, $workstr, 0,0);                 0520
               *badids* ← NULL;                                   0543
               ckidlist($workstr, $badids, idfno);                0523
               IF badids.L THEN                                   0521
                  BEGIN                                           0524
                  setigrps( idcrec, $workstr, 0,0);               0534
                  *badids* ← "Bad Ident(s) in GROUPS field
                  (removed): ", *badids*;                         0525
                  dismes ( 2, $badids);                           0522
                  END;                                            0526
            %secondary org%                                       0485
               getisorg ( idcrec, $workstr, 0,0);                 0535
               *badids* ← NULL;                                   0544
               ckidlist($workstr, $badids, idfno);                0536
               IF badids.L THEN                                   0537
                  BEGIN                                           0538
                  setisorg( idcrec, $workstr, 0,0);               0539
                  *badids* ← "Bad Ident(s) in SECONDARY
                  ORGANIZATIONS field (removed): ", *badids*;      0540
                  dismes ( 2, $badids);                           0541
                  END:                                            0542
            END                                                   0516
         ELSE  %group/org%                                        0517
            BEGIN                                                 0518
            %membership%                                          0481
               getimem ( idcrec, $workstr, 0,0);                  0551
               *badids* ← NULL;                                   0552
               ckidlist($workstr, $badids, idfno);                0553
               IF badids.L THEN                                   0554
                  BEGIN                                           0555
                  setimem( idcrec, $workstr, 0,0);                0556
                  *badids* ← "Bad Ident(s) in MEMBERSHIP field
                  (removed): ", *badids*;                         0557
                  dismes ( 2, $badids);                           0558
                  END;                                            0559
            END;                                                  0519
         %subcollections%                                         0482
            getimem ( idcrec, $workstr, 0,0);                     0560
            *badids* ← NULL;                                      0561
            ckidlist($workstr, $badids, idfno);                   0562
            IF badids.L THEN                                      0563
               BEGIN                                              0564
```

```
                    setimem( idcrec, $workstr, 0,0);                    0565
                    *badids* ← "Bad Ident=(s) in MEMBERSHIP field
                    (removed): ", *badids*;                             0566
                    dismes ( 2, $badids);                               0567
                    END;                                                0568
        IF nwrecflag THEN                                               060
           IF upidfil(idcident, idcrec, idfno, now) THEN                062
              dismes(2, $"Completed.")                                  063
           ELSE                                                         064
              dismes(2, $"Unsuccessful! The ident has already been used
              by another user.  Please respecify ident and update
              again.")                                                  065
        ELSE                                                            066
           IF modidfil(idcident, idcrec, idfno, now) THEN               067
              dismes(2, $"Completed.")                                  068
           ELSE                                                         069
              dismes(2, $"Unsuccessful! Illegal ident or record
              format.");                                                070
        END;                                                            071
   ENDCASE;                                                             072
   RETURN( &resultptr );                                                073
                                                                        074
   END.
                                                                        075

(xistatus)      % Show Staxus of record %
PROCEDURE (resultptr, parsemode, identptr, fieldname);                  076
   LOCAL rectype;                                                      0469
   LOCAL TEXT POINTER z1;                                              0466
   LOCAL STRING idstring[500], recstring[2000], string[2000];           078
   REF resultptr, identptr, fieldname:                                  079
   CASE parsemode OF                                                    080
      = parsing:                                                        081
         BEGIN                                                          082
         IF identptr = $loaded THEN                                    0455
            BEGIN                                                      0896
            IF NOT idcrec OR NOT [idcrec].L THEN err($"no record
            loaded!");                                                 0897
            idstatus ( idcident, idcrec, idcrtype, $string );           089
            fbctl(typecalit, $string):                                 0907
            END                                                        0898
         ELSE                                                          0456
            BEGIN                                                      0457
            z1 ← identptr; z1[1] ← identptr[d2sel+1];                  0464
            *idstring* ← identptr z1;                                  0465
            IF NOT ckident($idstring, $recstring, idfno) THEN          0459
               err($"Illegal Ident");                                  0460
            CASE fieldname OF                                          0590
               =$all:                                                  0591
                  BEGIN                                                0612
                  rectype ← IF jgrptst($recstring, 0) THEN grptyp      0467
                     ELSE IF orgtst($recstring, 0) THEN orgtyp ELSE
                     indtyp;                                           0468
                  idstatus ( $idstring, $recstring, rectype, $string);
                                                                       0463
                  fbctl(typecalit, $string);                            090
                  END;                                                 0613
```

```
                ENDCASE %use shoifield%                                    0592
                   BEGIN                                                   0593
                   *string* ← *[idcrec]*;                                  0598
                   *[idcrec]* ← *recstring*;                               0599
                   *recstring* ← *string*;                                 0600
                   *string* ← *[idcident]*;                                0601
                   *[idcident]* ← *idstring*;                              0602
                   *idstring* ← *string*;                                  0603
                   ON SIGNAL ELSE                                          0597
                      BEGIN                                                0610
                      *[idcrec]* ← *recstring*;                            0607
                      *[idcident]* ← *idstring*;                           0608
                      END;                                                 0609
                   shoifield (&resultptr, parsemode, &fieldname);          0604
                   *[idcrec]* ← *recstring*;                               0605
                   *[idcident]* ← *idstring*;                              0606
                   END;                                                    0594
             END;                                                          0458
          END;                                                             092
       = backup, = cleanup: shoifield(&resultptr, parsemode, 0);           0869
       ENDCASE;                                                            093
    RETURN( &resultptr );                                                  094
    END.                                                                   095

(newrec)        %initialize current record, for adding new records to
ident-file%
PROCEDURE (resultptr, parsemode);                                          096
    REF resultptr;                                                         098
    CASE parsemode OF                                                      099
       = parsing:                                                          0100
          BEGIN                                                            0101
          nwrecflag ← idmodflag ← idmodified ← TRUE;                       0102
          *[idcrec]* ← "()


          ";                                                               0103
          *[idcident]* ← NULL;                                             0104
          END;                                                             0105
    ENDCASE;                                                               0106
    RETURN( &resultptr );                                                  0107
    END.                                                                   0108

(isnewrec)      %is the current record a new record%
PROCEDURE (resultptr, parsemode);                                          0109
    REF resultptr;                                                         0111
    CASE parsemode OF                                                      0112
       = parsing:                                                          0117
          RETURN( IF nwrecflag THEN &resultptr ELSE 0 );                   0114
    ENDCASE;                                                               0115
    RETURN( &resultptr );                                                  0116
    END.                                                                   0117

(canmodify)     %is the user allowed to modify this record?%
PROCEDURE (resultptr, parsemode);                                          0118
    REF resultptr;                                                         0120
    CASE parsemode OF                                                      0121
```

```
      = parsing:                                                        0122
         BEGIN                                                          0893
         IF identwheel OR idmodflag THEN RETURN( &resultptr );          0123
         err($"You are not allowed to modify this record!");            0812
         END;                                                           0894
   ENDCASE;                                                             0124
   RETURN( &resultptr );                                                0125
   END.
                                                                        0126
(anychanges)   %has the user made any changes to the current record%
PROCEDURE (resultptr, parsemode);                                       0853
   REF resultptr;                                                       0855
   CASE parsemode OF                                                    0856
      = parsing:                                                        0857
         RETURN( IF idmodflag AND idmodified THEN &resultptr ELSE 0  );
                                                                        0858
   ENDCASE;                                                             0859
   RETURN( &resultptr );                                                0860
   END.
                                                                        0861
(iwheel)       %is the user an ident-wheel%
PROCEDURE (resultptr, parsemode);                                       0802
   REF resultptr;                                                       0804
   CASE parsemode OF                                                    0805
      = parsing:                                                        0806
         RETURN( IF identwheel THEN &resultptr ELSE 0  );               0807
   ENDCASE;                                                             0808
   RETURN( &resultptr );                                                0809
   END.
                                                                        0810
(asstentid)    %assign tentative ident to new record%
PROCEDURE (resultptr, parsemode);                                       0127
   LOCAL char, i;                                                       0661
   LOCAL STRING newid[50], namestr[150];                                0644
   REF resultptr;                                                       0129
   CASE parsemode OF                                                    0130
      = parsing:                                                        0131
         BEGIN  %generate an ident from the name field%                 0132
         IF NOT idcrec OR NOT [idcrec].L THEN err($"no record loaded");
                                                                        0899
         CASE idcrtype OF %assign first try%                            0616
            = orgtyp:  *newid* ← "AFF-1";                               0645
            = grptyp:                                                   0646
               BEGIN                                                    0654
               getinam(idcrec, $namestr, 0,0);                          0656
               IF NOT namestr.L THEN                                    0668
                  err($"name field must be specified before Ident can be
                  assigned");                                           0669
               makgid($namestr, $newid);                                0652
               END;                                                     0653
            = indtyp:                                                   0647
               BEGIN                                                    0648
               getifnf(idcrec, $namestr, 0,0);                          0651
               IF NOT namestr.L THEN                                    0671
                  err($"name field must be specified before Ident can be
                  assigned");                                           0672
```

```
            FIND SF(*namestr*);                                          0655
            *newid* ← READC;                                             0657
            LOOP CASE char ← READC OF                                    0658
               = SP:                                                     0659
                  BEGIN                                                  0664
                  CASE char ← READC OF                                   0675
                     = PT:    *newid* ← *newid*, char;                   0676
                     = ENDCHR: EXIT LOOP;                                0677
                     ENDCASE REPEAT CASE;                                0678
                  END;                                                   0665
               = '-: *newid* ← *newid*, char;                            0662
               = ENDCHR: EXIT LOOP;                                      0674
               ENDCASE;                                                  0660
            END;                                                         0649
         ENDCASE                                                         0650
            err ($"Illegal record type discovered in ASSTENTID");
                                                                         0673
      FOR i←0 UP UNTIL >= 9 DO %now check uniguenesss and modify by
      adding digits%                                                     0618
         IF oldid($newid, idfno) THEN                                    0621
            gnxtid($newid, idfno)                                        0622
         ELSE EXIT LOOP;                                                 0667
      setiid(idcrec, $newid);                                            0133
      */idcident/* ← *newid*;                                           0134
      END;                                                               0136
   ENDCASE;                                                              0137
   RETURN( &resultptr );                                                 0138
   END.
                                                                         0139
(noident)      %is there no ident assigned to current record%
PROCEDURE (resultptr, parsemode);                                        0140
   REF resultptr;                                                        0142
   CASE parsemode OF                                                     0143
      = parsing:                                                         0144
         RETURN( IF /idcident/.L THEN &resultptr ELSE 0 );               0145
   ENDCASE;                                                              0146
   RETURN( &resultptr );                                                 0147
   END.
                                                                         0148
(isindividual) %is the current record for an individual%
PROCEDURE (resultptr, parsemode);                                        0163
   REF resultptr;                                                        0165
   CASE parsemode OF                                                     0166
      = parsing:                                                         0167
         BEGIN                                                           0900
         IF NOT idcrec OR NOT /idcrec/.L THEN err($"no record loaded");
                                                                         0901
         RETURN( IF idcrtype = indtyp THEN &resultptr ELSE FALSE );      0168
         END;                                                            0903
   ENDCASE;                                                              0169
   RETURN( &resultptr );                                                 0170
   END.
                                                                         0171
(okident)      %is the passed ident ok in syntax and uniqueness%
PROCEDURE (resultptr, parsemode, identstr);                              0172
   REF resultptr, identstr;                                              0174
```

```
CH1,   2-OCT-74 07:29          < NLS, PSIDENT.NLS;4, /  0

   CASE parsemode OF                                                    0175
      = parsing:                                                        0176
         RETURN( IF ( FIND SF(*[idcident]*) -(LD/'-) ) OR
         oldid(idcident, idfno) THEN 0 ELSE &resultptr );               0177
   ENDCASE;                                                             0178
   RETURN( &resultptr );                                                0179
   END.
                                                                        0180
(checkname)    %check the ident file for individuals with this last
name, print brief status for each%
PROCEDURE (resultptr, parsemode, lastname);                             0181
   LOCAL TEXT POINTER z1;                                               0885
   LOCAL STRING idstr[50], namestr[50];                                 0886
   REF resultptr, lastname;                                             0183
   CASE parsemode OF                                                    0184
      = parsing:                                                        0185
         BEGIN                                                          0888
         z1 ← lastname[d2sel]; z1[1] ← lastname[d2sel+1];               0889
         *namestr* ← lastname z1;                                       0890
         RETURN( IF namesearch ($namestr, $idstr, lname, idfno) THEN
         &resultptr ELSE 0 );                                           0186
         END;                                                           0891
   ENDCASE;                                                             0187
   RETURN( &resultptr );                                                0188
   END.
                                                                        0189
(xiaddmem)     % add members to membership list %
PROCEDURE (resultptr, parsemode, fieldname, idlist);                    0190
   REF resultptr, fieldname, idlist;                                    0192
   LOCAL TEXT POINTER z1, z2, z3, z4;                                   0689
   LOCAL STRING                                                         0687
      workstr[500], idstring[50], newidlist[500],  COMMSTR(200];        0706
   CASE parsemode OF                                                    0193
       = parsing:                                                       0194
         BEGIN                                                          0195
         IF NOT idcrec OR NOT [idcrec].L THEN err($"no recod loaded");
                                                                        0904
         CASE fieldname OF                                              0682
            = $membership: getimem (idcrec, $workstr, 0,0);             0683
            = $groups: getigrps(idcrec, $workstr, 0,0);                 0684
            ENDCASE                                                     0685
               err($"Illegal field type encountered by XIADDMEM");      0688
         z1 ← idlist[d2sel]; z1[1] ← idlist[d2sel+1];                   0690
         *newidlist* ← idlist z1;                                       0691
         FIND SF(*newidlist*) ↑z4;                                      0692
         LOOP                                                           0693
            BEGIN                                                       0694
            FIND z4 $(SP/',) ↑z1 $(LD/'-) ↑z2 $SP ('( ↑z3 ←z3
            (')/ENDCHR] ↑z4 / ↑z3 ↑z4);                                 0695
            *idstring* ← +z1 z2;                                        0696
            *commstr* ← z3 z4;                                          0705
            IF NOT idstring.L THEN EXIT LOOP;                           0697
            FIND SF(*workstr*) ↑z1;                                     0700
            WHILE ( FIND z1 [*idstring*] ↑z1 ) DO                       0698
               IF FIND z1 < [(SP/',) > CH /ENDCHR] *idstring*
               (SP/',/ENDCHR) THEN                                      0701
```

```
                REPEAT LOOP 2;                                          0702
            *workstr* ← *idstring*, *commstr*, SP, *workstr*;           0703
            END;                                                        0704
        CASE fieldname OF                                               0707
            = $membership: setimem (idcrec, $workstr, 0,0);             0708
            = $groups: setigrps(idcrec, $workstr, 0,0);                 0709
            ENDCASE                                                     0710
                err($"Illegal field type encountered by XIADDMEM");     0711
        idmodified ← TRUE;                                              0862
        END;                                                            0681
    ENDCASE;                                                            0196
    RETURN( &resultptr );                                               0197
    END.                                                                0198

(xidelmem)      % delete members to membership list %
PROCEDURE (resultptr, parsemode, fieldname, idlist);                    0199
    REF resultptr, fieldname, idlist;                                   0713
    LOCAL TEXT POINTER z1, z2, z3, z4;                                  0714
    LOCAL STRING                                                        0715
        workstr[500], idstring[50], newidlist[500];                     0716
    CASE parsemode OF                                                   0717
        = parsing:                                                      0718
        BEGIN                                                           0719
        IF NOT idcrec OR NOT [idcrec].L THEN err($"no record loaded");  0905
        CASE fieldname OF                                               0720
            = $membership: getimem (idcrec, $workstr, 0,0);             0721
            = $groups: getigrps(idcrec, $workstr, 0,0);                 0722
            ENDCASE                                                     0723
                err($"Illegal field type encountered by XIADDMEM");     0724
        z1 ← idlist[d2sel]; z1[1] ← idlist[d2sel+1];                    0725
        *newidlist* ← idlist z1;                                        0726
        FIND SF(*newidlist*) ↑z2:                                       0727
        LOOP                                                            0728
            BEGIN                                                       0729
            FIND z2 $(SP/',) ↑z1 $(LD/'-) ↑z2;                          0730
            *idstring* ← +z1 z2;                                        0731
            IF NOT idstring.L THEN EXIT LOOP;                           0733
            FIND SF(*workstr*) ↑z4;                                     0734
            WHILE ( FIND z4 [*idstring*] ↑z4 ) DO                       0735
                IF FIND z4 < [(SP/',) > CH /ENDCHR] ↑z3 *idstring*
                ($(SP/',)/ENDCHR) ('( ['))/ENDCHR] / ) ↑z4 THEN         0736
                    BEGIN                                               0750
                    ST z3 z4 ← NULL;                                    0751
                    REPEAT LOOP 2;                                      0737
                    END;                                                0749
            *idstring* ← "Ident not found: ", *idstring*;               0753
            dismes(2, $idstring);                                       0752
            END;                                                        0739
        CASE fieldname OF                                               0740
            = $membership: setimem (idcrec, $workstr, 0,0);             0741
            = $groups: setigrps(idcrec, $workstr, 0,0);                 0742
            ENDCASE                                                     0743
                err($"Illegal field type encountered by XIADDMEM");     0744
        idmodified ← TRUE;                                              0863
        END;                                                            0745
```

```
    ENDCASE;                                                        0746
    RETURN( &resultptr );                                           0747
    END.
                                                                    0748
(xidelete)      % delete members to membership list %
PROCEDURE (resultptr, parsemode, recfield, which);                  0208
    LOCAL proc, rectype, param[4]; REF proc;                        0210
    LOCAL TEXT POINTER z2;                                          0765
    LOCAL STRING  idstring[50], recstring[2000], workstr[100];      0761
    REF resultptr, recfield, which;                                 0211
    CASE parsemode OF                                               0212
       = parsing:                                                   0213
          BEGIN                                                     0864
          CASE recfield OF                                          0214
             = $record:                                             0215
                BEGIN %add to delete group membership%              0216
                %check ident%                                       0787
                   z2 ← which[d2sel];                               0788
                   z2[1] ← which[d2sel+1];                          0789
                   *workstr* ← which z2;                            0790
                   IF NOT oldid($workstr, idfno) THEN               0791
                      err ($"Illegal Ident specified");             0792
                %save contents of current record%                   0782
                   *idstring*+ ← *[idcident]*;                       0755
                   *recstring* ← *[idcrec]*;                        0756
                   rectype ← idcrtype;                              0757
                %load the record "DELETE"%                          0783
                   *workstr* ← "DELETE";                            0773
                   FIND SF(*workstr*) ↑param SE(*workstr*) ↑z2;     0774
                   param[2] ← z2;  param[3] ← z2[1];                0775
                   xiloadrecord (&resultptr, parsemode, $param);    0776
                %add new ident to membership list%                  0784
                   param ← $membership;                             0777
                   xiaddmem (&resultptr, parsemode, $param, &which); 0778
                %update the ident file%                             0785
                   param ← TRUE;                                    0779
                   xiupdate (&resultptr, parsemode, $param);        0780
                dismes(2, $"Use EXPUNGE command to actually delete the
                record for this ident");                            0781
                %restore the old record%                            0786
                   *[idcident]* ← *idstring*;                       0770
                   *[idcrec]* ← *recstring*;                        0771
                   idcrtype ← rectype;                              0772
                END;                                                0218
             = $field:                                              0219
                BEGIN                                               0220
                IF NOT idcrec OR NOT [idcrec].L THEN err($"no record
                loaded");                                           0906
                CASE which OF                                       0221
                   = $function: &proc ← $setifunction;              0222
                   = $comments: &proc ← $setimcmnts;                0223
                   = $phone: &proc ← $setiphone;                    0224
                   = $secondary: %organization% &proc ← $setisorg;  0225
                   = $ nls: %mail address%                          0226
                      BEGIN                                         0227
                      setiuser ( idcrec, $nullfield, 0,0);          0228
```

```
                    &proc ← $setinlhost;                                    0229
                    END;                                                    0230
                = $network: %mail address%                                  0231
                    BEGIN                                                   0232
                    setinma(idcrec, $nullfield, 0,0);                       0233
                    &proc ← $setihost;                                      0234
                    END;                                                    0235
                = $hardcopy: %mail address% &proc ← setiadd;                0236
                = $subcollections: &proc ← $setisubcol;                     0237
                ENDCASE err(notyet);                                        0238
            proc (idcrec, $nullfield, 0,0);                                 0239
            END;                                                            0240
        ENDCASE err(notyet);                                                0241
    idmodified ← TRUE;                                                      0865
    END;                                                                    0866
ENDCASE;                                                                    0242
RETURN( &resultptr );                                                       0243
END.
                                                                            0244
(xiverfil)      %verify the contents of the Master Ident-File%
PROCEDURE (resultptr, parsemode, what, errorstop);                          0245
    LOCAL which, contonerror;                                               0247
    LOCAL TEXT POINTER z1;                                                  0794
    LOCAL STRING idstring[50];                                              0793
    REF resultptr, what, errorstop;                                         0248
    CASE parsemode OF                                                       0249
        = parsing:                                                          0250
            BEGIN                                                           0251
            *idstring* ← NULL;                                              0252
            which ← filver;                                                 0253
            CASE what OF                                                    0254
                = $everything: which ← -1;                                  0255
                = $individual: which ← which .V inds;                       0256
                = $used: %idents% which ← which .V usedids;                 0257
                = $group: which ← which .V groups;                          0258
                = $organization: %idents% which ← which .V orgs;            0259
                ENDCASE %an ident%                                          0260
                    BEGIN                                                   0796
                    z1 ← what[d2sel];  z1[1] ← what[d2sel+1];               0797
                    *idstring* ← what  z1;                                  0795
                    END;                                                    0799
            CASE errorstop OF                                               0261
                = $yes: contonerror ← TRUE;                                 0262
                ENDCASE contonerror ← FALSE;                                0263
            dismes(1, $"Verifying Master Ident-File.  Please wait.");       0264
            veridfile (idfno, which, contonerror, $idstring);               0265
            dismes(2, $"Completed.");                                       0266
            END;                                                            0267
        ENDCASE;                                                            0268
    RETURN( &resultptr );                                                   0269
    END.
                                                                            0270
(xipassword)    % check ident password %
PROCEDURE (resultptr, parsemode, param);                                    0271
    LOCAL TEXT POINTER z1;                                                  0273
    LOCAL STRING string[20];                                                0274
```

```
   REF resultptr, param;                                                0275
   CASE parsemode OF                                                    0276
      = parsing:                                                        0277
         BEGIN                                                          0278
         zl ← param[d2sel]; zl[1] ← param[d2sel+1];                     0279
         *string* ← +param zl;                                          0280
         IF *string* = "RABBIT" THEN RETURN( &resultptr);               0281
         IF NOT identwheel THEN                                         0800
            err($"Illegal password supplied.  Your attempt to use this
            priviledged command has been reported to system personell");
                                                                        0801
         err($"Illegal password");                                      0282
         END;                                                           0283
   ENDCASE;                                                             0284
   RETURN( &resultptr );                                                0285
   END.
                                                                        0286
(xiloadrecord) % load an old record %
PROCEDURE (resultptr, parsemode, param);                                0287
   LOCAL TEXT POINTER zl;                                               0289
   LOCAL STRING idstring[30];                                           0290
   REF resultptr, param;                                                0291
   CASE parsemode OF                                                    0292
      = parsing:                                                        0293
         BEGIN                                                          0294
         zl ← param;  zl[1] ← param[d2sel+1];                           0295
         *idstring* ← param zl;                                         0296
         *[idcrec]* ← NULL;                                             0297
         IF NOT ckident($idstring, idcrec, idfno) THEN                  0298
            err($"Illegal Ident");                                      0299
         *[idcident]* ← NULL;                                           0300
         getiid(idcrec, idcident, 0,0);                                 0301
         nwrecflag ← FALSE;                                             0302
         IF jgrptst(idcrec, 0) THEN idcrtype ← grptyp                   0303
         ELSE                                                           0304
            IF orgtst(idcrec, 0) THEN idcrtype ← orgtyp                 0305
            ELSE idcrtype ← indtyp;                                     0306
         CASE idcrtype OF                                               0813
            = indtyp: %individual%                                      0814
               idmodflag ← IF *[idcident]* = *initsr* THEN TRUE ELSE
               FALSE;                                                   0815
            ENDCASE                                                     0816
               BEGIN                                                    0817
               geticord (idcrec, $idstring, 0,0);                       0818
               idmodflag ← IF *idstring* = *initsr* THEN TRUE ELSE
               FALSE;                                                   0819
               END;                                                     0820
         idmodified ← FALSE;                                            0867
         END;                                                           0307
   ENDCASE;                                                             0308
   RETURN( &resultptr );                                                0309
   END.
                                                                        0310
(setifield)    % set specified fieldname to specified value %
PROCEDURE (resultptr, parsemode, fieldname, value, value2, value3);     0311
   LOCAL proced;                                                        0313
```

```
LOCAL TEXT POINTER t1, t2, t3;                                          0314
LOCAL STRING                                                            0315
   locstr[200];  %for passing strings to core routines%                 0316
REF resultptr, fieldname, proced, value, value2, value3;                0317
CASE parsemode OF                                                       0318
   = parsing:                                                           0319
      BEGIN                                                             0320
      %check for valid record loaded%                                   0321
         IF NOT idcrec OR NOT [idcrec].L THEN                           0322
            err($"Use LOAD RECORD or ADD RECORD command before
            changing/setting fields");                                  0323
      t1 ← value[d2sel];  t1[1] ← value[d2sel+1];                       0324
      CASE fieldname OF                                                 0325
         =$approve:                                                     0326
            BEGIN                                                       0327
            setiverify(idcrec, $"Verified", 0, 0);                      0328
            RETURN( &resultptr );                                       0329
            END;                                                        0330
         =$capabilities:  &proced ← $seticapability;                    0331
         =$coordinator:                                                 0332
            IF idcrtype = indtyp THEN                                   0333
               err($"Illegal for an INDIVIDUAL's record")               0334
            ELSE &proced ← $seticord;                                   0335
         =$comment:  &proced ← $setimcmnts;                             0336
         =$delivery:                                                    0337
            BEGIN                                                       0823
            &proced ← $setidelivery;                                    0822
            CASE value OF                                               0826
               = $nls: *locstr* ← "NLS";                                0827
               = $network: *locstr* ← "Network";                        0828
               ENDCASE *locstr* ← "Hardcopy";                           0829
            FIND SF(*locstr*) ↑value SE(*locstr*) ↑ t1;                 0830
            END;                                                        0824
         =$expand:                                                      0338
            BEGIN                                                       0339
            IF idcrtype = indtyp THEN                                   0340
               err($"Illegal for an INDIVIDUAL's record");              0341
            &proced ← $setiexp;                                         0342
            CASE value OF                                               0343
               = $yes: *locstr* ← "Expand";                             0344
               = $no: *locstr* ← NULL;                                  0345
               ENDCASE err($"Illegal value for expand field");          0346
            FIND SF(*locstr*) ↑value SE(*locstr*) ↑ t1;                 0347
            END;                                                        0348
         =$function: &proced ← $setifun;                                0349
         =$groups:  &proced ← $setigrps;                                0350
         =$ident:                                                       0351
            BEGIN                                                       0352
            *[idcident]* ← value t1;                                    0353
            IF nwrecflag THEN  &proced ← $setiid                        0354
            ELSE RETURN ( &resultptr );                                 0355
            END;                                                        0356
         =$usadd: %U.S. Mail address%  &proced ← $setiadd;              0357
         =$netadd: %Network Mail address%  &proced ← $setinma;          0358
         =$nethost: %Network Mail address%  &proced ← $setihost;        0359
         =$nlsadd: %NLS Mail address%  &proced ← $setiuser;             0360
```

```
=$nlshost: %NLS Mail address%  &proced ← $setinlhost;
=$name:
   BEGIN
   &proced ← $setinam;
   IF idcrtype = indtyp THEN %must concatenate names
   togeter%
      BEGIN
      t2 ← value2[d2sel];  t2[1] ← value2[d2sel+1];
      t3 ← value3[d2sel];  t3[1] ← value3[d2sel+1];
      *locstr* ← value t1. ", ", value2 t2, SP, value3 t3;

      stnamcap ($locstr );
      FIND SF(*locstr*) ↑value SE(*locstr*) ↑t1;
      END
   END;
=$membership:  &proced ← $setimem;
=$organization:
   BEGIN
   IF idcrtype NOT= indtyp THEN err($"Only allowed for
   INDIVIDUAL's records");
   IF value = $independent THEN
      BEGIN
      *locstr* ← "IND";
      FIND SF(*locstr*) ↑value SE(*locstr*) ↑t1;
      END;
   &proced ← $setiorg;
   END;
=$phone:  &proced ← $setiphone;
=$rtype: %record type%
   BEGIN
   idcrtype ← (CASE value OF
      = $group: grptyp;
      = $individual: indtyp;
      = $organization: orgtyp;
      ENDCASE err(notyet) );
   RETURN ( &resultptr );
   END;
=$secondary: %organization%
   IF idcrtype NOT= indtyp THEN
      err($"Only allowed for INDIVIDUAL's records")
   ELSE &proced ← $setisorg;
=$subcollections:  &proced ← $setisubcol;
=$type: %of organization%
   IF idcrtype NOT= orgtyp THEN err($"Only allowed for
   ORGANIZATION's records")
   ELSE
      BEGIN
      &proced ← $setitype;
      CASE value OF
         = $independent: *locstr* ← "Independent";
         = $user: *locstr* ← "User";
         = $server: *locstr* ← "Server";
         = $tip: *locstr* ← "Tip";
         = $associate: *locstr* ← "Associate";
         ENDCASE err($"Illegal Organization type");
      FIND SF(*locstr*) ↑value SE(*locstr*) ↑t1;
```

```
            END;                                                    0844
         ENDCASE err(notyet);                                       0392
      %call appropriate setj procedure%                             0393
         *locstr* ← Value t1;                                       0394
         proced( idcrec, $locstr, 0,0 );                            0395
      idmodified ← TRUE;                                            0868
      END;                                                          0396
   ENDCASE;                                                         0397
RETURN( &resultptr );                                               0398
END.
                                                                    0399
(shoifield)    % show the specified fieldname of current record %
PROCEDURE (resultptr, parsemode, fieldname);                        0400
   LOCAL string, proced;                                            0402
   REF string;                                                      0870
   LOCAL TEXT POINTER z1;                                           0892
   REF resultptr, fieldname, proced;                                0405
   CASE parsemode OF                                                0406
      = parsing:                                                    0407
         BEGIN                                                      0408
         &proced ← 0;                                               0883
         IF NOT idcrec OR NOT /idcrec/.L THEN                       0409
            err($"Use LOAD RECORD or ADD RECORD command before
            changing/setting fields");                              0410
         %get string from storage allocator%                        0872
            IF NOT &string ← resultptr/4/ ← getstring ( 1000, $dsppblk)
            THEN                                                    0873
               err($"Storage allocator is out of free space");      0876
         CASE fieldname OF                                          0411
            =$approve: &proced ← $getiverify;                       0412
            =$capabilities: &proced ← $geticapability;              0413
            =$coordinator: &proced ← $geticord;                     0414
            =$comment: &proced ← $getimcmnts;                       0415
            =$delivery: &proced ← $getidelivery;                    0416
            =$expand: &proced ← $getiexp;                           0417
            =$function: &proced ← $getifun;                         0418
            =$groups: &proced ← $getigrps;                          0419
            =$ident:                                                0420
               BEGIN                                                0421
               *string* ← */idcident/*;                             0422
               &proced ← 0;                                         0881
               END;                                                 0424
            =$usadd: %U.S. Mail address%  &proced ← $getiadd;       0425
            =$netadd: %Network Mail address%  &proced ← $getinma;   0426
            =$nethost: %Network Mail address%  &proced ← $getihost; 0427
            =$nlsadd: %NLS Mail address%  &proced ← $getiuser;      0428
            =$nlshost: %NLS Mail address%  &proced ← $getinlhost;   0429
            =$name: &proced ← $getinam;                             0430
            =$membership: &proced ← $getimem;                       0431
            =$organization:  &proced ← $getiorg;                    0432
            =$phone: &proced ← $getiphone;                          0433
            =$rtype: %record type%                                  0434
               BEGIN                                                0435
               CASE idcrtype OF                                     0436
                  = indtyp: *string* ← "group";                     0437
                  = grptyp: *string* ← "individual";                0438
```

```
                = orgtyp: *string* ← "organization";                    0439
                ENDCASE err(notyet) ;                                   0440
             &proced ← 0;                                               0441
             END;                                                       0442
          =$secondary: %organization%  &proced ← $getisorg;             0443
          =$subcollections: &proced ← $getisubcol;                      0444
          =$type: %of organization%  &proced ← $getitype;               0445
          ENDCASE err(notyet);                                          0446
       %call appropriate getj procedure%                                0447
          IF &proced THEN                                               0884
             IF NOT proced( idcrec, &string, 0,0 ) THEN *string* ←
             "NONE";                                                    0448
       FIND SF(*string*) ↑resultptr SE(*string*) ↑z1;                   0879
       resultptr[2] ← z1;  resultptr[3] ← z1[1];                        0880
       IF string.L THEN                                                 0908
          fbctl(typecalit, &string)                                     0910
       ELSE                                                             0909
          fbctl(typecalit, $"[NONE]");                                  0449
       END;                                                             0450
    = backup, = cleanup: IF resultptr[4] THEN freestring(resultptr[4]
    := 0, sdspblk);                                                     0871
    ENDCASE;                                                            0451
 RETURN( &resultptr );                                                  0452
 END.
                                                                        0453
FINISH                                                                  0454
```

PS PROGS

(MLK)PSPROGS (MLK)PSPROGS (MLK)PSPROGS (MLK)PSPROGS (MLK)PSPROGS

```
< NLS, PSPROGS.NLS;10, >, 4-OCT-74 14:50 KEV ;;;;
FILE psprogs % L10 <rel-nls>psprogs %% (110,) (rel-nls,psprogs.rel,) %
                                                                        02
% DECLARATIONS %                                                       03
   DEFINE                                                               04
   prsavesize = 30#,    % enough for 10 replace parse rules %           05
   prrecsize = 3#,      % wordsize of prrecord %                        06
   sbstacksize = 20#,   % size of sbstack %                            07
   sbentsize =2#;       % size of sbentry records %                     08
% PROGRAMS SUBSYSTEM %                                                  09
   (xpattach)           % attach subsystem to NLS %                    0685
      PROCEDURE( resultptr, parsemode, subsysptr,dispptr);              0686
      REF resultptr, subsysptr, dispptr;                                0687
      % --------------- %                                              0688
      CASE parsemode OF                                                 0689
         = parsing:                                                     0690
            BEGIN                                                       0691
            % define the subsystem given the specified handle %        0692
               dfnsubsys( dispptr, gtctrlbits(dispptr), $nlssubs);      0693
            END;                                                        0694
         ENDCASE;                                                       0695
      RETURN( &resultptr );                                             0696
      END.
                                                                        0697
   (xpcompile) PROCEDURE        % compile program %                    0545
      %FORMALS%                                                         0546
         (resultptr,    %result record pointer%                        0547
         parsemode,     %parsing mode%                                 0548
         entity,        %entity command word%                          0549
         location       %selected location to begin%,                  0550
         compiler,      %name of compiler%                             0551
         fname);        %nam of output file%                           0552
      LOCAL                                                             0553
         adstr[40],     % data structure for link parsing %             0554
         savecsp,       % save location for current csp %               0555
         errors,        % error count %                                 0556
         tptr2,         % local ptr to text ptr %                       0557
         da;            % ptr to display area %                         0558
      LOCAL STRING                                                      0559
         errstr[50],    % error string %                                0560
         rfilname[200];         % output rel-file name %                0561
      REF resultptr, da, entity, location, compiler, fname, tptr2;      0562
      CASE parsemode OF                                                 0563
         = parsing:                                                     0564
            BEGIN                                                       0565
            &da ← lda();                                                0566
            CASE entity OF                                              0567
               = $file: % compile file %                                0568
                  BEGIN                                                 0569
                  % put file names into local strings %                 0570
                     lnkprs( &compiler, $adstr);                        0571
                     CASE lnpfls( &fname, 0, $rfilname) OF              0572
                        = lhostn: NULL;                                 0573
                        ENDCASE                                         0574
                           err($"Remote File Manipulations Not
                           Implemented Yet");                           0575
```

```
                % set dacsp in the display area record and invoke the
                compiler %                                              0576
                   savecsp ← da.dacsp;                                  0577
                   da.dacsp ← location;                                 0578
                   errors ← cpcmpfl( FALSE, $adstr, $rfilname, &da);
                                                                        0579
                   da.dacsp ← savecsp;                                  0580
                % tell the user about any errors %                      0581
                   IF errors > 0 THEN                                   0582
                      BEGIN                                             0583
                      *errstr* ← STRING(errors), " error(s):  Type
                      CA.";                                             0584
                      dismes (1, $errstr);                              0585
                      clrbuf (0);   % clear the input buffer %          0586
                      LOOP IF inpcuc() = CA THEN EXIT;                  0587
                      dismes (0);                                       0588
                      END;                                              0589
                ENDCASE err(notyet);                                    0590
                END;                                                    0591
             = $llo:  % LLO User Program %                              0592
                BEGIN                                                   0593
                gpllo( location, &da );                                 0594
                END;                                                    0595
             = $procedure:  % Compile procedure %                       0596
                BEGIN                                                   0597
                cpcmproc( location, &da );                              0598
                END;                                                    0599
             = $content:  % analyzer filter %                           0600
                BEGIN                                                   0601
                da.dacacode ← cpconan(&location, &da);                  0602
                da.davspec.vscapf ← TRUE;                               0603
                END;                                                    0604
          ENDCASE err(notyet);                                          0605
          END;                                                          0606
   ENDCASE;                                                             0607
   RETURN( &resultptr );
   END.                                                                 0608

(xpdelete)           % Delete a user program from stack %               093
   PROCEDURE (resultptr, parsemode, howmany );                          094
   REF resultptr, howmany;                                              095
   CASE parsemode OF                                                    096
      = parsing:                                                        097
         CASE howmany OF                                                098
            = $all:  % all programs in the stack %                      099
               BEGIN                                                    0711
               gpgmrst();                                               0100
               IF upgbsz > $upgbdf THEN gpbsz($upgbdf);                 0714
               END;                                                     0713
            = $last:  % only the last program in the stack %            0101
               gppop();                                                 0102
         ENDCASE err(notyet);                                           0103
   ENDCASE;                                                             0104
   RETURN (&resultptr);                                                 0105
   END.
                                                                        0106
(xpdeinstitute)      % Deinstitute a user program %                     0107
```

```
   PROCEDURE (resultptr, parsemode, type);                                  0108
   LOCAL t2ptr, da, field;                                                  0109
   LOCAL STRING nmstr[50];                                                  0110
   REF resultptr, type, t2ptr, da;                                          0111
   CASE parsemode OF                                                        0112
      = parsing:                                                            0113
         BEGIN                                                              0114
         % decode type to determine field %                                 0115
            field ←                                                         0116
               CASE type OF                                                 0117
                  = $content:       % content analyzer program %           0118
                     dacacode;                                              0119
                  = $seqgenerator:  % sequence generator program %
                                                                            0120
                     dausqcod;                                              0121
                  = $sort:          % sort key program %                   0122
                     daukeycod;                                             0123
               ENDCASE err(notyet);                                         0124
         % deinstitute program from current display area %                 0125
            &da ← lda();                                                    0126
            da.field ← 0;                                                   0127
         % make sure we dont undo this in cmdfinish %                      0543
            cspupdate ← FALSE;                                              0544
         END;                                                               0128
   ENDCASE;                                                                 0129
   RETURN (&resultptr);                                                     0130
   END.
                                                                            0131
(xpdetach)          % detach subsystem to NLS %                           0698
   PROCEDURE( resultptr, parsemode, subsysptr,dispptr);                     0699
   REF resultptr, subsysptr, dispptr;                                       0700
   % ---------------- %                                                    0701
   CASE parsemode OF                                                        0702
      = parsing:                                                            0703
         BEGIN                                                              0704
         % detach the subsystem given the specified handle %               0705
            delsubsys( dispptr, $nlssubs);                                  0706
         END;                                                               0707
      ENDCASE;                                                              0708
   RETURN( &resultptr );                                                    0709
   END.
                                                                            0710
(xpinstitute)       % Institute a user program %                           0156
   PROCEDURE (resultptr, parsemode, pgmnam, type);                          0157
   LOCAL t2ptr, field, da;                                                  0158
   LOCAL STRING nmstr[50];                                                  0159
   REF resultptr, pgmnam, t2ptr, type, da;                                  0160
   CASE parsemode OF                                                        0161
      = parsing:                                                            0162
         BEGIN                                                              0163
         % move program name into local string %                           0164
            &t2ptr ← &pgmnam + d2sel;                                       0165
            *nmstr* ← pgmnam t2ptr;                                         0166
         % decode type to determine field %                                 0167
            field ←                                                         0168
               CASE type OF                                                 0169
```

```
                    = $content:        % content analyzer program % 0170
                       dacacode;                                   0171
                    = $seqgenerator:   % sequence generator program %
                                                                   0172
                       dausqcod;                                   0173
                    = $sort:           % sort key program %         0174
                       daukeycod;                                  0175
                 ENDCASE err(notyet);                              0176
              % institute program into current display area %      0177
                 &da ← lda();                                      0178
                 da.field ← upgenv( $nmstr );                      0179
              % make sure we dont undo this in cmdfinish %         0541
                 cspupdate ← FALSE;                                0542
              END;                                                 0180
        ENDCASE;                                                   0181
        RETURN (&resultptr);                                       0182
        END.
                                                                   0183
(xpkill)              % kill tenex subsystem %                     0184
        PROCEDURE (resultptr, parsemode);                          0185
        REF resultptr;                                             0186
        CASE parsemode OF                                          0187
           = parsing:                                              0188
              BEGIN                                                0189
              % kill the tenex subsystem %                        0190
                 gpkill();                                         0191
              END;                                                 0192
           ENDCASE;                                                0193
        RETURN (&resultptr);                                       0194
        END.
                                                                   0195
        % %                                                       0196
```

```
(xpload)              % load a user program %                          0652
   PROCEDURE (resultptr, parsemode, filnm);                           0653
   LOCAL adstr[40];                                                   0654
   REF resultptr, filnm;                                              0655
   cspupdate ← FALSE;                                                 0656
   CASE parsemode OF                                                  0657
      = parsing:                                                      0658
         BEGIN                                                        0659
         % parse file name link %                                     0660
            lnkprs( &filnm, $adstr);                                  0661
         % load the program into the user programs buffer %           0662
            gpget( $adstr, TRUE % display messages in gpget % );      0663
         END;                                                         0664
   ENDCASE;                                                           0665
   RETURN (&resultptr);                                               0666
   END.
                                                                      0667
(xprun)               % Run a user program %                          0221
   PROCEDURE (resultptr, parsemode, pgmnam);                          0222
   LOCAL t2ptr;                                                       0223
   LOCAL STRING prog[50];                                             0224
   REF resultptr, pgmnam, t2ptr;                                      0225
   CASE parsemode OF                                                  0226
      = parsing:                                                      0227
         BEGIN                                                        0228
         % move filename to local string %                           0229
            &t2ptr ← &pgmnam + d2sel;                                 0230
            *prog* ← pgmnam t2ptr;                                    0231
         % call the program %                                        0232
            gpexpgm( $prog );                                         0233
         END;                                                         0234
   ENDCASE;                                                           0235
   RETURN (&resultptr);                                               0236
   END.
                                                                      0237
(xpreplace)                    % Replace Parserule command %          0238
   PROCEDURE (resultptr, parsemode, oldrule, filenam,  NEWRULE<;      0239
   LOCAL tp2; REF tp2;                                                0240
   LOCAL instloc, ptr, newloc;  REF instloc, ptr, newloc;             0241
   LOCAL STRING locstr[200];                                          0244
   LOCAL STRING errstr[75];                                           0245
   REF resultptr, oldrule, filenam, newrule;                          0246
   % ---------------- %                                              0247
   CASE parsemode OF                                                  0248
      = parsing:                                                      0249
         BEGIN                                                        0250
         % mark the NDDT symbol table to use symbols in the named
         file as locals %                                             0251
            resultptr ← FALSE; % in case we have an error %           0252
            % move file name to local string %                       0253
               CASE lnbfls( &filenam, 0, $locstr) OF                  0528
                  = lhostn: NULL;                                     0529
                  ENDCASE                                             0530
                     err($"Remote File Manipulations Not Implemented
                     Yet");                                           0531
            ddtmark( $locstr );                                       0263
```

```
            resultptr ← 1; % so we will pop the symbol table again %
         % look up the oldrule %
            &tp2 ← &oldrule + d2sel;
            *locstr* ← oldrule tp2;
            IF NOT ddtlookup($locstr, FALSE: &instloc) THEN
               BEGIN
               (errout):
               *errstr* ← *locstr*, " could not be found";
               err( $errstr );
               END;
         % reset the symbol table mark %
            resultptr ← FALSE;
            ddtpop();
         % lookup the new rule %
            &tp2 ← &newrule + d2sel;
            *locstr* ← newrule tp2;
            IF NOT ddtlookup($locstr, FALSE: &newloc) THEN
               GOTO errout;
         % set ptr to point to a slot in the prsavearea %
            FOR &ptr ← $prsavearea UP prrecsize UNTIL >=
            $prsavearea+prsavx DO
               IF NOT ptr.prexists THEN GOTO buildentry;
         % allocate a new entry at the end of the stack if there is
         room %
            IF prsavx + prrecsize > prsavesize
               THEN err($"Too many parse rules replaced, must reset
               some first");
            &ptr ← $prsavearea + prsavx;
            prsavx ← prsavx + prrecsize;
         % initialize a new entry %
            (buildentry):
            ptr.prwrd1 ← instloc;
            ptr.prwrd2 ← instloc[1];
            ptr.praddr ← &instloc;
            ptr.prexists ← TRUE;
         % replace the first instruction by an "EXECUTE newrule"
         instruction %
            instloc.nsuccessor ← 0;
            instloc.opcode ← 22; %*** EXECUTE ***%
            instloc.addr ← &newloc;
         END;
      = backup, = cleanup:
         IF resultptr THEN ddtpop();
   ENDCASE;
   RETURN (&resultptr);
   END.

(xpreset)            % Reset command %
   PROCEDURE (resultptr, parsemode, entity, rulename, fname);
   LOCAL instptr, ptr, tp2;
   REF resultptr, entity, rulename, fname, instptr, ptr, tp2;
   LOCAL STRING locstr[200], errstr[50];
   % ---------------- %
   CASE parsemode OF
      = parsing:
```

```
        BEGIN                                                          0316
        resultptr ← FALSE;                                             0317
        CASE entity OF                                                 0318
           = $ouffer: % set buffer size %                              0319
             gpbsz( 4 ); % set buffer size to 4 pages (default) %
                                                                       0320
           = $nddt:  % NDDT control-n %                                0321
             nddtdisarm();                                             0322
           = $parserule:                                               0323
             BEGIN                                                     0324
             % mark the NDDT symbol table to use symbols in the
             named file as locals %                                    0325
                resultptr ← FALSE; % in case we have an error %
                                                                       0326
                % move file name to local string %                     0327
                   CASE lnbfls( &fname, 0, $locstr) OF                 0524
                      = lhostn: NULL;                                  0525
                      ENDCASE                                          0526
                         err($"Remote File Manipulations Not
                         Implemented Yet");                            0527
                ddtmark( $locstr );                                    0337
                resultpt° ← 1; % so we will pop the symbol table
                again %                                                0338
             % look up the rulename %                                  0339
                &tp2 ← &rulename + d2sel;                              0340
                *locstr* ← rulename tp2;                               0341
                IF NOT ddtlookup($locstr, FALSE: &instptr) THEN
                                                                       0342
                   BEGIN                                               0343
                   *errstr* ← *locstr*, " could not be found";         0344
                   err( $errstr );                                     0345
                   END;                                                0346
             % reset the symbol table mark %                           0347
                resultptr ← FALSE;                                     0348
                ddtpop();                                              0349
             % find the entry in the prsavearea corresponding to
             the rule name  %                                          0350
                FOR &ptr ← $prsavearea UP prrecsize UNTIL >=
                $prsavearea+prsavx DO                                  0351
                   IF ptr.praddr THEN                                  0352
                      BEGIN                                            0353
                      % reset the instruction %                        0354
                         instptr ← ptr.prwrd1;                         0355
                         instptr[1] ← ptr.prwrd2;                      0356
                         ptr.prexists ← FALSE;                         0357
                         RETURN( &resultptr );                         0358
                      END;                                             0359
             % didn't find the named rule in the prsavearea, tell
             the user %                                                0360
                err( $"named rule not in the replace stack" );         0361
             END;                                                      0362
          ENDCASE err(notyet);                                         0363
        END;                                                           0364
     = backup, = cleanup:                                              0365
        IF resultptr THEN ddtpop();                                    0366
  ENDCASE;                                                             0367
```

```
    RETURN (&resultptr);                                                0368
    END.
                                                                        0369
(xprunt)            % Run tenex subsystem %                             0610
    PROCEDURE                                                           0611
        ( resultptr,                                                    0612
        parsemode,                                                      0613
        subnam,         % name of subsystem to be run %                  0614
        outfil,         % FALSE for tty output; else output file name %
                                                                        0615
        inmode,         % 3-file; 1-interactive; 2-typeahead; 4-none %
                                                                        0616
        intext,         % file name / typeahead / termination character
        %                                                               0617
        wtmode          % TRUE for wait for completion %                0618
        );                                                              0619
    LOCAL t2ptr, rhostn, rhost2, adstr[40];                             0620
    LOCAL STRING typ[2000], outjfn[200];                                0621
    REF resultptr, subnam, outfil, inmode, intext, t2ptr, wtmode;       0622
                                                                        0623
    CASE parsemode OF                                                   0624
        = parsing:                                                      0625
            BEGIN                                                       0626
            % parse subsystem name link %                               0627
              lnkprs( &subnam, $adstr);                                 0628
            % move output file name to local string %                   0629
              rhostn ← lhostn;                                          0715
              IF outfil THEN                                            0630
                 rhostn ← lnbfls( &outfil, 0, $outjfn);                 0631
            % move filename/typeahead/termination char to local string%
                                                                        0632
              rhost2 ← lhostn;                                          0716
              CASE inmode OF                                            0633
                 = 1, = 2:    % typeahead or interactive mode %         0634
                   BEGIN                                                0635
                   &t2ptr ← &intext + d2sel;                            0636
                   *typ* ← intext t2ptr;                                0637
                   END;                                                 0638
                 = 3:          % from file %                           0639
                   rhost2 ← lnbfls( &intext, 0, $typ);                  0640
                 = 4:  NULL;  % no input %                              0641
                 ENDCASE err($"Illegal Input Mode");                    0642
            % handle any error conditions %                             0643
              ON SIGNAL ELSE gpkill();                                  0644
            % go do the work %                                          0645
              gprunt( $adstr, rhostn, $outjfn, inmode, rhost2, $typ,
              wtmode);                                                  0646
            END;                                                        0647
        ENDCASE;                                                        0648
    RETURN (&resultptr);                                                0649
    END.
                                                                        0650
    % %                                                                 0651
```

```
(xpset)              % Set command %                                    0438
   PROCEDURE (resultptr, parsemode, entity, param);                    0439
   LOCAL tptr2, i, size;                                               0440
   LOCAL STRING sizestr[20];                                           0441
   REF resultptr, entity, param, tptr2;                                0442
   CASE parsemode OF                                                   0443
      = parsing:                                                       0444
         CASE entity OF                                                0445
            = $buffer: % set buffer size %                             0446
               BEGIN                                                   0447
               % move size string into local string, convert to a
               numeric value %                                         0448
                  &tptr2 ← &param + d2sel;                             0449
                  *sizestr* ← param tptr2;                             0450
                  size ← VALUE( $sizestr );                            0451
               % set the buffer to the new size %                      0452
                  IF NOT gpbsz( size ) THEN                            0453
                     err($"Invalid size for programs buffer");         0454
               END;                                                    0455
            = $nddt:  % arm NDDT control-h %                           0456
               BEGIN                                                   0457
               nddtarm();                                              0458
               END;                                                    0459
         ENDCASE err(notyet);                                          0460
   ENDCASE;                                                            0461
   RETURN (&resultptr);                                                0462
   END.
                                                                       0463
(xpsetf)             %set up globals for alternate directory in lsel%
                                                                       0668
   PROCEDURE (resultptr, parsemode);                                   0669
   REF resultptr;                                                      0670
   CASE parsemode OF                                                   0671
      = parsing:                                                       0672
         BEGIN                                                         0673
         %set flag indicating presence of alteratte directory%         0674
            altdfl ← TRUE;                                             0675
         %set up alternate directory and alternate extension
         strings%                                                      0676
            *altdir* ← "PROGRAMS,"; %to add more separate by commas%
                                                                       0677
            *altext* ← "REL,CA,SK,SG,SUBSYS,CML,PROC-REP,";            0678
         END;                                                          0679
      = cleanup: %reset the global flag%                               0680
         altdfl ← FALSE;                                               0681
   ENDCASE;                                                            0682
   RETURN (&resultptr);                                                0683
   END.
                                                                       0684
(xpshow)             % Show Status of user programs display %          0464
   PROCEDURE (resultptr, parsemode);                                   0465
   LOCAL da;                                                           0466
   LOCAL STRING statusstr[400];                                        0467
   REF resultptr, da;                                                  0468
   CASE parsemode OF                                                   0469
      = parsing:                                                       0470
```

```
            BEGIN                                                       0471
            &da ← lda();                                                0472
            gpstatus( $statusstr, &da);                                 0473
            fbctl( typecalit, $statusstr );                             0474
            END;                                                        0475
      ENDCASE;                                                          0476
      RETURN (&resultptr);                                              0477
      END.
                                                                        0478
   (xpshtn)              % Show Tenex Subsystem Status display %        0479
      PROCEDURE (resultptr, parsemode);                                 0480
      LOCAL STRING statusstr[400];                                      0481
      REF resultptr;                                                    0482
      CASE parsemode OF                                                 0483
         = parsing:                                                     0484
            BEGIN                                                       0485
            IF gptxst( infork, $statusstr ) THEN                        0486
               fbctl( typecalit, $statusstr )                           0487
            ELSE fbctl( typecalit, $"No Tenex Subsystem Running");      0488
            END;                                                        0489
      ENDCASE;                                                          0490
      RETURN (&resultptr);                                              0491
      END.
                                                                        0492
FINISH of psprogs                                                       0493
```

PSREADMAIL

PSREADMAIL PSREADMAIL PSREADMAIL PSREADMAIL PSREADMAIL PSREADM
PSREADMAIL PSREADMAIL PSREADMAIL PSREADMAIL PSREADMAIL PSREADM

```
( NLS, PSREADMAIL.NLS;1, ), 22-MAR-74 05:41 KEV ;
FILE psreadmail % L10 <rel-nls>psreadmail %% (L10,)
(rel-nls,psreadmail.rel,) %                                          02
% READMAIL SUBSYTEM %
                                                                     03
   (xjaccept)          % accept the specified authors/titlewords/dates
   for the filter through which mail is passed to the user %         04
      PROCEDURE (resultptr, parsemode);                              05
      REF resultptr;                                                 06
      CASE parsemode OF                                              07
         = parsing: NULL;                                            08
      ENDCASE;                                                       09
      RETURN( &resultptr );                                          010
      END.
                                                                     011
   (xjdelitem)                % delete the specified item from the
   current catagory %                                                012
      PROCEDURE (resultptr, parsemode, jitem);                       013
      REF resultptr, jitem;                                          014
      CASE parsemode OF                                              015
         = parsing: NULL;                                            016
      ENDCASE;                                                       017
      RETURN( &resultptr );                                          018
      END.
                                                                     019
   (xjsetcurcat)       % set current catagory %                       020
      PROCEDURE (resultptr, parsemode, catagory);                    021
      REF resultptr, catagory;                                       022
      CASE parsemode OF                                              023
         = parsing: NULL;                                            024
      ENDCASE;                                                       025
      RETURN( &resultptr );                                          026
      END.
                                                                     027
   (xjprint)           % print specified item (or ALL of a catagory) on
   the line-printer %                                                028
      PROCEDURE (resultptr, parsemode, jitem=/:
      ref resultptr, jitem;                                          030
      CASE parsemode OF                                              031
         = parsing: NULL;                                            032
      ENDCASE;                                                       033
      RETURN( &resultptr );                                          034
      END.
                                                                     035
   (xjgetcuritem)      % get the current item number  -- IS THIS
   NEEDED?? %                                                        036
      PROCEDURE (resultptr, parsemode);                              037
      REF resultptr:                                                 038
      CASE parsemode OF                                              039
         = parsing: NULL;                                            040
      ENDCASE;                                                       041
      RETURN( &resultptr );                                          042
      END.
                                                                     043
   (xjshwitem)         % Show the specified item %                    044
      PROCEDURE (resultptr, parsemode);                              045
```

```
   REF resultptr;                                                    046
   CASE parsemode OF                                                 047
      = parsing: NULL;                                               048
   ENDCASE;                                                          049
   RETURN( &resultptr );                                             050
   END.
                                                                     051
(xjshwcat)          % Show all the items in the current catagory %
                                                                     052
   PROCEDURE (resultptr, parsemode);                                 053
   REF resultptr;                                                    054
   CASE parsemode OF                                                 055
      = parsing: NULL;                                               056
   ENDCASE;                                                          057
   RETURN( &resultptr );                                             058
   END.
                                                                     059
(xjfileitem)        % file the specified item under the specified
catagory %                                                           060
   PROCEDURE (resultptr, parsemode, jitem, catagory);                061
   REF resultptr, jitem, catagory;                                   062
   CASE parsemode OF                                                 063
      = parsing: NULL;                                               064
   ENDCASE;                                                          065
   RETURN( &resultptr );                                             066
   END.
                                                                     067
(xjlast)             / TRUE if last current item is last item in
current catagory %                                                   068
   PROCEDURE (resultptr, parsemode);                                 069
   REF resultptr;                                                    070
   CASE parsemode OF                                                 071
      = parsing: RETURN(FALSE);                                      072
   ENDCASE;                                                          073
   RETURN( &resultptr );                                             074
   END.
                                                                     075
(xjomit)            % omit specified authors/titlewords/dates from
user printing of journal items %                                     076
   PROCEDURE (resultptr, parsemode, type, list);                     077
   REF resultptr, type, list;                                        078
   CASE parsemode OF                                                 079
      = parsing: NULL;                                               080
   ENDCASE;                                                          081
   RETURN( &resultptr );                                             082
   END.
                                                                     083
(xjremind)          % remind user about this item at specified time %
                                                                     084
   PROCEDURE (resultptr, parsemode, jitem  when);                    085
   REF resultptr, jitem, when;                                       086
   CASE parsemode OF                                                 087
      = parsing: NULL;                                               088
   ENDCASE;                                                          089
   RETURN( &resultptr );                                             090
   END.
```

```
                                                                    091
(xjexpunge)                 % expunge deleted items from mail
catagories %                                                        092
   PROCEDURE (resultptr, parsemode);                                093
   REF resultptr;                                                   094
   CASE parsemode OF                                                095
        parsing: NULL;                                              096
   ENDCASE;                                                         097
   RETURN( &resultptr );                                            098
   END.
                                                                    099
FINISH of PSREADMAIL                                                0100
```

PSSENDMAIL

(MLK)PSSENDMAIL (MLK)PSSENDMAIL (MLK)PSSENDMAIL (MLK)PSSENDMAIL (
(MLK)PSSENDMAIL (MLK)PSSENDMAIL (MLK)PSSENDMAIL (MLK)PSSENDMAIL (

```
< NLS, PSSENDMAIL.NLS;4, >, 4-OCT-74 13:31 CHI ;;;;< NLS,
PSSENDMAIL.NLS;4, >, 25-JUN-74 11:54 CHI ;
FILE pssendmail % L10 to <rel-nls>pssendmail.rel % % (110.sav,)
(rel-nls, pssendmail.rel=@) %                                          02
DECLARE EXTERNAL STRING %for send mail status and forms%              0725
   sjnumber = "NUMBER:",                                              0726
   sjtitle = "TITLE:",                                                0727
   sjcomment = "COMMENT:",                                            0730
   sjauthor = "AUTHOR(S):",                                           0731
   sjaction = "DISTRIBUTE FOR ACTION TO:",                            0732
   sjinfo = "DISTRIBUTE FOR INFO-ONLY TO:",                           0733
   sjsubcol = "SUBCOLLECTION(S):",                                    0728
   sjkeywords = "KEYWORD(S):",                                        0729
   sjhandling = "HANDLING INSTRUCTION:",                              0734
   sjrecording = "RECORDING INSTRUCTION:",                            0736
   sjhardcopy = "OFFLINE ITEM -- LOCATED AT:",                        0737
   sjrfc = "RFC NUMBER:",                                             0738
   sjobsoletes = "OBSOLETES ITEM NUMBER(S):",                         0735
   sjaccess = "ACCESS STATUS:",                                       0739
   sjupdates = "UPDATE TO ITEM NUMBER(S):",                           0740
   sjlink = "INSERT LINK TO FOLLOW:",                                 0741
   sjforward = "FORWARD ITEM NUMBER:",                                0742
   sjmessage = "MESSAGE:",                                            0749
   sjbranch = "BRANCH AT:",                                           0743
   sjplex = "PLEX AT:",                                               0744
   sjgroup = "GROUP AT:",                                             0745
   sjfile = "FILE:",                                                  0746
   sjsendit = "SEND THE MAIL.",                                       0747
   sjbacksendit = ".LIAM EHT DNES";                                   0748
% SENDMAIL SUBSYTEM %
                                                                        03
   (xjzapworfil)       % initialize work file %                        012
      PROCEDURE (resultptr, parsemode);                                013
      REF resultptr;                                                   014
      LOCAL STRING locstr[200];                                        015
      CASE parsemode OF                                                016
         = parsing:                                                    017
            BEGIN                                                      018
            IF jworkstid THEN resetf(jworkstid.stfile)                 019
            ELSE                                                       020
               BEGIN                                                   021
               *locstr* ← "[JWORK-", *initsr*, "].SYSTEM;";            022
               jworkstid ← openwk(0, $locstr);                         023
               END;                                                    024
            FIND SF(jworkstid) ↑jwpl;                                  025
            initjwork();                                              0255
            END;                                                       026
      ENDCASE;                                                         027
      RETURN( &resultptr );                                            028
      END.
                                                                       029
   (xjloaworfil)       % load Journal work file %
   PROCEDURE (resultptr, parsemode);  %returns TRUE if old file being
   used, FAISE if new file created and initialized%                    030
      REF resultptr;                                                   032
      LOCAL STRING locstr[200];                                        033
```

```
   CASE parsemode OF                                                    034
      = parsing:                                                        035
         BEGIN                                                         0716
         IF NOT jworkstid.stfile THEN                                  0268
            BEGIN                                                      036
            *locstr* ← '<,  *userstr*, '>, "[SEND-MAIL]."  *initsr*,
            ";1;P707000";                                              037
            ON SIGNAL ELSE                                             038
               BEGIN %must create it%                                  039
               ON SIGNAL ELSE;                                         040
               jworkstid ← openwk(0, $locstr);                         041
               GOTO ok;                                               0262
               END;                                                    044
            jworkstid ← orgstid;                                       045
            jworkstid.stfile ← open(0, $locstr);                       046
            (ok):  %finishup and leave%                            0263
            FIND SF(jworkstid) ↑jwpl;                                  047
            [fintadr(jworkstid.stfile)].finoclos ← TRUE;           0264
            END;                                                       048
         initjwork()                                                  0716
         END;                                                         0717
   ENDCASE;                                                            049
   RETURN( &resultptr );                                               050
   END.
                                                                       051
(xjcloworfil)        % close work file %                               052
   PROCEDURE (resultptr, parsemode);                                   053
   REF resultptr;                                                      054
   CASE parsemode OF                                                   055
      = parsing:                                                       056
         BEGIN                                                         057
         IF jworkstid.stfile THEN                                     0266
            [fintadr(jworkstid.stfile)].finoclos ← FALSE;            0267
         jworkstid ← 0;                                              059
         END;                                                        060
      ENDCASE;                                                       061
   RETURN( &resultptr );                                             062
   END.
                                                                      063
(xjreserve)                    % reserve numbers %                   064
   PROCEDURE (resultptr, parsemode, type, numb, insertloc);          065
   LOCAL TEXT POINTER z1, z2;                                       0431
   LOCAL str;  REF str;                                             0395
   REF resultptr, type, numb, insertloc;                             066
   CASE parsemode OF                                                 067
      = parsing:                                                     068
         BEGIN                                                       0393
         IF NOT &str ← getstring(200, $dspblk) THEN                  0663
            err($"Storage allocator out of space");                  0664
         getnums (&str, $initsr, type, IF NOT numb THEN 1 ELSE
         getpint(&numb, &numb+d2sel), 1);                            0394
         FIND SF(*str*) ↑z1 SE(*str*) ↑z2;                           0426
         IF insertloc THEN %insertloc it as a visible%               0424
            BEGIN                                                    0425
            cinstex (&insertloc, $z1, $z2, TRUE);                    0427
            END                                                      0428
```

```
        ELSE %just show him%                                          0429
           fbctl (typecalit, &str);                                   0430
        resultptr ← z1; resultptr[1] ← z1[1];                         0604
        resultptr[d2sel] ← z2; resultptr[d2sel+1] ← z2[1];            0605
        resultptr[4] ← &str;                                          0665
        END;                                                          0396
     = backup, = cleanup: IF resultptr[4] THEN                        0666
        freestring(resultptr[4] := 0, $dspblk);                       0667
     ENDCASE;                                                          069
  RETURN( &resultptr );                                                070
  END.
                                                                       071
(xjrfcreserve)      % reserve an RFC and a Catalog numstr %
PROCEDURE (resultptr, parsemode, titlestr, authstr, sendto,
onlineflag, insertloc);                                               072
  LOCAL TEXT POINTER z1, z2;                                          0406
  LOCAL STRING                                                        0407
     auth[50], titl[200], send[150], rfcnum[50], numstr[100];         0411
  REF resultptr, titlestr, authstr, sendto, onlineflag, insertloc;
                                                                      074
  CASE parsemode OF                                                    075
     = parsing:                                                        076
        BEGIN                                                        0396
        %build strings for rfcex%                                    0423
           z2 ← authstr[d2sel]; z2[1] ← authstr[d2sel+1];            0406
           *auth* ← authstr z2;                                      0415
              IF NOT auth?L THEN *auth* ← *initsr*;                  0606
           z2 ← sendto[d2sel]; z2[1] ← sendto[d2sel+1];              0416
           *send* ← sendto z2;                                       0417
           z2 ← titlestr[d2sel]; z2[1] ← titlestr[d2sel+1];          0412
           *titl* ← titlestr z2;                                     0418
        rfcex($auth, $titl, $send, onlineflag, $rfcnum, $numstr);
                                                                     0400
        *numstr* ← "RFC numstr: ", *rfcnum*, ", Catalog numstr: ",
        *numstr*;                                                    0405
        IF insertloc THEN %insertloc it as a visible%                0402
           BEGIN                                                     0419
           FIND SF(*numstr*) ↑z1 SE(*numstr*) ↑z2;                   0420
           cinstex (&insertloc, $z1, $z2, TRUE);                     0421
           END                                                       0422
        ELSE %just show him%                                         0403
           fbctl (typecalit, $numstr);                               0401
        END;                                                         0399
  ENDCASE;                                                           077
  RETURN( &resultptr );                                              078
  END.
                                                                     079
(xjrfcsho)          % show status of RFC reserve requestr %
PROCEDURE (resultptr, parsemode, titlestr, authstr, sendto,
onlineflag);                                                         0432
  LOCAL TEXT POINTER aptr, tptr, dptr;                               0433
  LOCAL STRING str[200];                                             0434
  REF resultptr, titlestr, authstr, sendto, onlineflag;              0436
  CASE parsemode OF                                                  0437
     = parsing:                                                      0438
        BEGIN                                                        0439
```

```
        %build text-pointers for concatinating status message%   0440
           aptr ← authstr[d2sel]; aptr[1] ← authstr[d2sel+1];    0441
           dptr ← sendto[d2sel]; dptr[1] ← sendto[d2sel+1];      0443
           tptr ← titlestr[d2sel]; tptr[1] ← titlestr[d2sel+1]; 0445
        *str* ←                                                  0448
           "Title: ", titlestr tptr, "
           Author(s): ", authstr aptr, "
           Send to: ", sendto dptr;                              0460
        IF onlineflag THEN                                       0461
           *str* ← *str*, "
           Online document"                                      0462
        ELSE                                                     0463
           *str* ← *str*, "
           Offline document";                                    0464
        %show him%                                               0454
           fbctl (typelit, $str);                                0455
        END;                                                     0456
   ENDCASE;                                                      0457
   RETURN( &resultptr );                                         0458
   END.                                                          0459

(xjdoit)              % finish submission or forward request %    088
   PROCEDURE (resultptr, parsemode);                             089
   REF resultptr;                                                090
   CASE parsemode OF                                             093
      = parsing:                                                 094
        BEGIN                                                    095
        jsubmit();                                               096
        END;                                                     097
   ENDCASE;                                                      098
   RETURN( &resultptr );                                         099
                                                                0100
   END.                                                         0101

(xjlock)              % lock or unlock the journal %             0689
   PROCEDURE (resultptr, parsemode, password, lockflag);         0690
   LOCAL TEXT POINTER z1, z2;                                    0691
   LOCAL STRING passstr[20];                                     0692
   REF resultptr, password, lockflag;                            0693
   CASE parsemode OF                                             0694
      = parsing:                                                 0695
        BEGIN                                                    0696
        % get information out of the parameter records %         0697
           z1 ← password;                                        0698
           z1[1] ← [&password + 1];                              0699
           z2 ← [&password + 2];                                 0700
           z2[1] ← [&password + 3];                              0701
           *passstr* ← z1 z2;                                    0702
        % Check the password value. %                            0703
           IF *passstr* # *jnlpsw* THEN                          0704
              BEGIN                                              0705
              dismes(1, $"Illegal password");                    0706
              EXIT CASE;                                         0707
              END;                                               0708
        % Lock or unlock the journal %                           0709
           IF lockflag THEN lockjo(0 %jlock flag%)               0710
```

```
            ELSE unlkjo(0);                                              0711
         END;                                                            0712
      ENDCASE;                                                           0713
   RETURN( ^resultptr );                                                 0714
   END.                                                                  0715

(xjpriharcop)        % print hard copy %                                  0110
   PROCEDURE (resultptr, parsemode);                                     0111
   REF resultptr;                                                        0112
   CASE parsemode OF                                                     0113
      = parsing: err(notyet);                                            0114
   ENDCASE;                                                              0115
   RETURN( &resultptr );                                                 0116
   END.                                                                  0117

(xjprocess)                    % process command form %                  0118
   PROCEDURE (resultptr, parsemode, destination);                        0119
   REF resultptr, destination;                                           0120
   LOCAL TEXT POINTER left, right, z1, z2;                               0284
   LOCAL STRING locstr[500];                                             0379
   CASE parsemode OF                                                     0121
      = parsing:                                                         0122
         BEGIN                                                           0343
         %Number%                                                        0372
            IF FIND SF(destination) [*sjnumber*] $SP ↑left [EOL] <
            CH $SP ↑right > THEN                                         0373
               BEGIN                                                     0377
               *locstr* ← left right;                                    0375
               IF locstr.L THEN setjnumber ($locstr)                     0376
               ELSE %assign number and update form%                      0380
                  BEGIN                                                  0381
                  gcatnums ($locstr, $initsr, $"JOURNAL", 1, 1);         0505
                  IF locstr.L THEN                                       0506
                     BEGIN                                               0511
                     setjnumber ($locstr);                               0509
                     ST left right ← *locstr*;                           0382
                     END;                                                0510
                  END;                                                   0383
               END;                                                      0378
         %title%                                                         0289
            IF FIND SF(destination) [*sjtitle*] $SP ↑left [EOL] < CH
            $SP ↑right > THEN                                            0345
               BEGIN                                                     0508
               *locstr* ← left right;                                    0290
               IF locstr.L THEN setjtitle ($locstr);                     0291
               END;                                                      0507
         %comment%                                                       0347
            IF FIND SF(destination) > [*sjcomment*] $SP ↑left [EOL]
            < CH $SP ↑right > THEN                                       0348
               BEGIN                                                     0512
               *locstr* ← left right;                                    0513
               IF locstr.L THEN setjcomment ($locstr);                   0514
               END;                                                      0515
         %author%                                                        0352
            IF FIND SF(destination) > [*sjauthor*] $SP ↑left [EOL] <
            CH $SP ↑right > THEN                                         0353
```

```
        BEGIN                                                   0516
        *locstr* ← left right; %supposealy an ident string%
                                                                0517
        IF locstr.L THEN setjauthor ($locstr);                  0518
        END;                                                    0519
    %Action distribution%                                       0357
       IF FIND SF(destination) > [*sjaction*] $SP ↑left [EOL] <
       CH $SP ↑right > THEN                                     0358
        BEGIN                                                   0520
        *locstr* ← left right; %supposedly an ident string%
                                                                0524
        IF locstr.L THEN setjaction ($locstr);                  0525
        END;                                                    0523
    %Information-only distribution%                             0362
       IF FIND SF(destination) > [*sjinfo*] $SP ↑left [EOL] <
       CH $SP ↑right > THEN                                     0363
        BEGIN                                                   0526
        *locstr* ← left right; %supposedly an ident string%
                                                                0530
        IF locstr.L THEN setjinfo ($locstr);                    0531
        END;                                                    0529
    %subcollections%                                            0367
       IF FIND SF(destination) > [*sjsubcol*] $SP ↑left [EOL] <
       CH $SP ↑right > THEN                                     0368
        BEGIN                                                   0532
        *locstr* ← left right; %supposedly an ident string%
                                                                0536
        IF locstr.L THEN setjsubcol ($locstr);                  0537
        END;                                                    0535
    %keywords%                                                  0465
       IF FIND SF(destination) > [*sjkeywords*] $SP ↑left [EOL]
       < CH $SP ↑right > THEN                                   0538
        BEGIN                                                   0539
        *locstr* ← left right;                                  0540
        IF locstr.L THEN setjkeyw ($locstr);                    0541
        END;                                                    0542
    %handling instructions%                                     0470
       IF FIND SF(destination) > [*sjhandling*] $SP ↑left [EOL]
       < CH $SP ↑right > THEN                                   0543
        BEGIN                                                   0544
        *locstr* ← left right;                                  0545
        IF locstr.L THEN setjexpedite (*locstr* =
        "Expedite");                                            0546
        END;                                                    0547
    %recording instructions%                                    0500
       IF FIND SF(destination) > [*sjrecording*] $SP ↑left
       [EOL] < CH $SP ↑right > THEN                             0548
        BEGIN                                                   0549
        *locstr* ← left right;                                  0550
        IF locstr.L THEN                                        0551
           setjexpedite (*locstr* = "Unrecorded");              0553
        END;                                                    0552
    %hardcopy location%                                         0475
       IF FIND SF(destination) > [*sjhardcopy*] $SP ↑left [EOL]
       < CH $SP ↑right > THEN                                   0554
        BEGIN                                                   0555
```

```
        *locstr* ← left right;                                          0556
        IF locstr.L THEN                                                0557
           BEGIN                                                        0561
           FIND SF(*locstr*) ↑left SE(*locstr*) ↑right;                 0559
           setjsource (hccpyv, $left, $right);                          0560
           END;                                                         0562
        END;                                                            0558
    %rfc number%                                                        0480
       IF FIND SF(destination) > [*sjrfc*] $SP ↑left [EOL] < CH
       $SP ↑right > THEN                                                0563
           BEGIN                                                        0564
           *locstr* ← left right;                                       0565
           IF locstr.L THEN setjrfc ($locstr);                          0566
           END;                                                         0571
    %obsoletes%                                                         0485
       IF FIND SF(destination) > [*sjobsoletes*] $SP ↑left
       [EOL] < CH $SP ↑right > THEN                                     0572
           BEGIN                                                        0573
           *locstr* ← left right;                                       0574
           IF locstr.L THEN setjobsoletes ($locstr);                    0575
           END;                                                         0576
    %access status%                                                     0678
       IF FIND SF(destination) > [*sjaccess*] $SP ↑left [EOL] <
       CH $SP ↑right > THEN                                             0679
           BEGIN                                                        0680
           *locstr* ← left right;                                       0681
           chprvsts (jwpl.stfile, (IF *locstr* = "PRIVATE"              0684
              THEN $psprivate                                           0687
              ELSE $pspublic));                                         0686
           END;                                                         0683
    %updates%                                                           0490
       IF FIND SF(destination) > [*sjupdates*] $SP ↑left [EOL]
       < CH $SP ↑right > THEN                                           0577
           BEGIN                                                        0578
           *locstr* ← left right;                                       0579
           IF locstr.L THEN setjupdates ($locstr);                      0580
           END;                                                         0581
    %insert link%                                                       0495
       IF FIND SF(destination) > [*sjlink*] $SP ↑left [EOL] <
       CH $SP ↑right > THEN                                             0582
           BEGIN                                                        0583
           *locstr* ← left right;                                       0584
           IF locstr.L THEN setjlink ($locstr);                         0585
           END;                                                         0586
    %message%                                                           0607
       IF FIND SF(destination) > [*sjmessage*] $SP ↑left [EOL]
       < CH $SP ↑right > THEN                                           0608
           BEGIN                                                        0609
           IF POS right > left THEN setjsource (stmtv, $left,
           $right);                                                     0611
           END;                                                         0612
    %brancn%                                                            0613
       IF FIND SF(destination) > [*sjbrancn*] $SP ↑left [EOL] <
       CH $SP ↑right > THEN                                             0614
           BEGIN                                                        0615
           z1 ← right;  z1[1] ← right[1];                               0860
```

```
            caddexp($left, $right, lda(), $z1);                          0618
            IF z1 NOT= endfil THEN setjsource (groupv, $z1, $z1);
                                                                         0619
            END;                                                         0617
         %plex%                                                          0626
            IF FIND SF(destination) > [*sjplex*] $SP ↑left [EOL] <
            CH $SP ↑right > THEN                                         0627
            BEGIN                                                        0628
            z1 ← right;  z1[1] ← right[1];                               0858
            caddexp($left, $right, lda(), $z1);                          0629
            IF z1 NOT= endfil THEN                                       0630
               BEGIN                                                     0642
               left ← gethed(z1);                                        0638
               right ← getail(z1);                                       0639
               left[1] ← right[1] ← 1;                                   0640
               setjsource (groupv, $left, $right);                       0641
               END;                                                      0643
            END;                                                         0631
         %group%                                                         0644
            IF FIND SF(destination) > [*sjgroup*] $SP ↑left [EOL] <
            CH $SP ↑right > THEN                                         0645
            BEGIN                                                        0646
            z1 ← right;  z1[1] ← right[1];                               0659
            caddexp($left, $right, lda(), $z1);                          0647
            IF z1 NOT= endfil THEN                                       0648
               IF FIND ↑right > [EOL] $SP ↑left [EOL] < CH $SP
               ↑right > THEN                                             0657
                  BEGIN                                                  0649
                  caddexp($left, $right, lda(), $z2);                    0658
                  grptst(z1, z2 : z1, z2);                               0656
                  z1[1] ← z2[1] ← 1;                                     0652
                  setjsource (groupv, $z1, $z2);                         0653
                  END;                                                   0654
            END;                                                         0655
         %file%                                                          0632
            IF FIND SF(destination) > [*sjfile*] $SP ↑left [EOL] <
            CH $SP ↑right > THEN                                         0633
            BEGIN                                                        0634
            IF POS right > left THEN setjsource (filev, $left,
            $right);                                                     0636
            END;                                                         0637
         %send it%                                                       0333
            IF FIND SE(destination) < *sjbacksendit* EOL THEN            0334
            jsubmit();                                                   0337
         END;                                                            0344
   ENDCASE;                                                              0123
   RETURN( &resultptr );                                                 0124
   END.
                                                                         0125
(xjinsstatus)        % Insert Status of submission form %                 0126
   PROCEDURE (resultptr, parsemode, destination, level);                 0127
   LOCAL TEXT POINTER z1, z2;                                            0600
   LOCAL STRING string[2000];                                            0261
   REF resultptr, destination, level;                                    0128
   CASE parsemode OF                                                     0129
      = parsing:                                                         0130
```

```
            BEGIN                                                   0257
            jstatus ($string);                                      0258
            FIND SF(*string*) ↑z1 SE(*string*) ↑z2;                 0259
            curmkr ← cinssta(destination, level, $z1, $z2);         0601
            dpset(dspstrc, curmkr, endfil, curmkr);                 0603
            curmkr[1] ← 1;                                          0602
            END;                                                    0260
      ENDCASE;                                                      0131
      RETURN( &resultptr );                                         0132
      END.
                                                                    0133
(xjstatus)              % Show Status of submission request %       0587
      PROCEDURE (resultptr, parsemode);                             0588
      LOCAL STRING string[2000];                                    0589
      REF resultptr;                                                0590
      CASE parsemode OF                                             0591
         = parsing:                                                 0592
            BEGIN                                                   0593
            jstatus ($string);                                      0594
            fbctl (typecalit, $string);                             0595
            END;                                                    0596
      ENDCASE;                                                      0597
      RETURN( &resultptr );                                         0598
      END.
                                                                    0599
(xjistatus)             % Show Status of ident record %
PROCEDURE (resultptr, parsemode, identptr, fieldname);              0750
      LOCAL rectype;                                                0751
      LOCAL TEXT POINTER z1;                                        0752
      LOCAL STRING idstring[500], recstring[2000], string[2000];    0753
      REF resultptr, identptr, fieldname;                           0754
      CASE parsemode OF                                             0755
         = parsing:                                                 0756
            BEGIN                                                   0757
            z1 ← identptr; z1[1] ← identptr[d2sel+1];               0758
            *idstring* ← identptr z1;                               0759
            IF NOT ckident($idstring, $recstring, idfno) THEN       0760
               err($"Illegal Ident");                               0761
            rectype ← IF jgrptst($recstring, 0) THEN grptyp         0762
               ELSE IF orgtst($recstring, 0) THEN orgtyp ELSE indtyp;
                                                                    0763
            jidstatus ( $idstring, $recstring, rectype, $string);   0764
            fbctl(typecalit, $string);                              0765
            END;                                                    0766
         = backup, = cleanup: IF resultptr[4] THEN
         freestring(resultptr[4] := 0, $dspblk);                    0767
         ENDCASE;                                                   0768
      RETURN( &resultptr );                                         0769
      END.
                                                                    0770
(xjforward)                     % Forward J doc to new ids %        0134
      PROCEDURE (resultptr, parsemode, jnumber, actionflag, ids);   0135
      REF resultptr, jnumber, actionflag, ids;                      0136
      LOCAL TEXT POINTER z2;                                        0864
      LOCAL STRING numberstr[20], idlst[300];                       0865
      CASE parsemode OF                                             0137
```

```
      = parsing:                                                    0138
         BEGIN                                                      0861
         FIND jnumber > $NP ↑jnumber $D ↑z2;                        0866
         *numberstr* ← jnumber z2;                                  0867
         z2 ← ids[d2sel]; z2[1] ← ids[d2sel+1];                     0868
         *idlst* ← ids z2;                                          0869
         secdist(&numberstr, $idlst, (IF actionflag = $action THEN
         TRUE ELSE FALSE));                                         0863
         END;                                                       0862
   ENDCASE;                                                         0139
   RETURN( &resultptr );                                            0140
   END.
                                                                    0141
(jsetfield)                    %set the specified field to the
specified value%                                                    0142
   PROCEDURE (resultptr, parsemode, field, value);                  0143
   LOCAL proced;                                                    0144
   LOCAL TEXT POINTER t1, t2;                                       0145
   LOCAL STRING
      locstr[1000];  %for passing strings to core routines%         0147
   REF resultptr, field, value, proced;                             0148
   CASE parsemode OF                                                0149
      = parsing:                                                    0150
         BEGIN                                                      0151
         t1 ← value[d2sel]; t1[1] ← value[d2sel+1];                 0152
         CASE field OF                                              0153
            =$action:  &proced ← $setjaction;                       0154
            =$authors:  &proced ← $setjauthor;                      0159
            =$branch, =$plex, =$group:                              0167
               BEGIN                                                0168
               setjsource(groupv, &value, $t1);                     0169
               RETURN (&resultptr);                                 0385
               END;                                                 0170
            =$comment:  &proced ← $setjcomment;                     0171
            =$expedite:                                             0176
               BEGIN                                                0177
               setjexpedite(value);                                 0178
               RETURN (&resultptr);                                 0386
               END;                                                 0179
            =$file: %number%                                        0180
               BEGIN                                                0181
               filnam(value.stfile, $locstr);                       0659
               FIND SF(*locstr*) ↑value SE(*locstr*) ↑t1;           0660
               setjsource(filev, &value, $t1);                      0662
               RETURN (&resultptr);                                 0387
               END;                                                 0183
            =$hardcopy:                                             0189
               BEGIN                                                0190
               setjsource(hcopyv, &value, $t1);                     0191
               RETURN (&resultptr);                                 0389
               END;                                                 0193
            =$information:  &proced ← $setjinfo;                    0194
            =$insert: %link%                                        0199
               BEGIN                                                0200
               setjlink($t1);                                       0201
               RETURN (&resultptr);                                 0390
```

```
            END;                                                    0202
          =$keywords:  &proced ← $setjkeyword;                      0203
          =$number:                                                 0206
            BEGIN                                                   0720
            *locstr* ← value t1;                                    0724
            FIND SF(*locstr*) ↑value $D ↑t1;                        0721
            &proced ← $setjnumber;                                  0719
            END;                                                    0723
          =$obsoletes:  &proced ← $setjobsoletes;                   0213
          =$private:                                                0668
            BEGIN                                                   0672
            chprvsts (jwpl.stfile, $psprivate);                     0669
            RETURN (&resultptr);                                    0677
            END;                                                    0675
          =$public:                                                 0670
            BEGIN                                                   0673
            chprvsts (jwpl.stfile, $pspublic);                      0671
            RETURN (&resultptr);                                    0676
            END;                                                    0674
          =$rfc:  &proced ← $setjrfc;                               0216
          =$statement:                                              0223
            BEGIN                                                   0224
            setjsource(stmtv, &value, $t1);                         0225
            RETURN (&resultptr);                                    0391
            END;                                                    0226
          =$subcollections:  &proced ← $setjsubcol;                 0227
          =$title:  &proced ← $setjtitle;                           0232
          =$unrecorded:                                             0237
            BEGIN                                                   0238
            setjunrecorded (TRUE);                                  0384
            RETURN (&resultptr);                                    0392
            END;                                                    0243
          =$update:  &proced ← $setjupdates;                        0244
          ENDCASE err(notyet);                                      0249
        %call appropriate setj procedure%                           0163
          *locstr* ← value t1;                                      0164
          proced( $locstr );                                        0165
        END;                                                        0250
      ENDCASE;                                                      0251
  RETURN( &resultptr );                                             0252
  END.
                                                                    0253

(jidstatus) %generate status string for an ident entry%
PROCEDURE (idstr, entrystr, typid, statstr);                        0771
  %This procedure inserts the status of the entry passed into the
  status string.%                                                   0772
  LOCAL TEXT POINTER ptr1, ptr2, ptr3, ptr4;                        0773
  LOCAL f1, i2;                                                     0774
  LOCAL STRING cadstr[300];                                         0775
  REF idstr, statstr;                                               0776
  *statstr* ← "Ident: ", *idstr*, EOL;                              0777
  getinam(entrystr, 0, $ptr1, $ptr2);                               0778
  *statstr* ← *statstr*, "Name: ", ptr1 ptr2, EOL;                  0779
  IF typid = indtyp THEN                                            0780
    BEGIN                                                           0781
    getiorg(entrystr, 0, $ptr1, $ptr2);                             0782
```

```
    *statstr* ← *statstr*, "Organization: ", ptr1 ptr2, EOL;          0783
    IF getisorg(entrystr, 0, $ptr1, $ptr2) THEN                       0784
       *statstr* ← *statstr*, "Secondary Organization(s): ",          0785
          ptr1 ptr2, EOL;                                             0786
    END                                                               0787
ELSE                                                                  0788
    BEGIN                                                             0789
    IF expdist(entrystr, 0) THEN                                      0790
       *statstr* ← *statstr*, "Expand", EOL                           0791
    ELSE *statstr* ← *statstr*, "Unexpanded", EOL;                    0792
    getimem(entrystr, 0, $ptr1, $ptr2);                               0793
    *statstr* ← *statstr*, "Membership: ", ptr1 ptr2, EOL;            0794
    geticord(entrystr, 0, $ptr1, $ptr2);                              0795
    *statstr* ← *statstr*, "Coordinator: ", ptr1 ptr2, EOL;           0796
    IF typid = orgtyp THEN                                            0797
       BEGIN                                                          0798
       getityp(entrystr, 0, $ptr1, $ptr2);                            0799
       *statstr* ← *statstr*, "Organization Type: ", ptr1 ptr2,
       EOL;                                                           0800
       END;                                                           0801
    END;                                                              0802
IF getigrps (entrystr, 0, $ptr1, $ptr2) THEN                          0803
    *statstr* ← *statstr*, "Groups: ", ptr1 ptr2, EOL;                0804
%Put Online and Network addresses in cadstr%                          0805
    *cadstr* ← NULL;                                                  0806
    f2 ← getinlhost(entrystr, 0, $ptr3, $ptr4);                       0807
    IF (f1 ← getiuser(entrystr, 0, $ptr1, $ptr2)) OR f2 THEN          0808
       BEGIN                                                          0809
       *cadstr* ← *cadstr*, "  Online(NLS)", EOL;                     0810
       IF f1 THEN *cadstr* ← *cadstr*, "    User: ", ptr1 ptr2,
       EOL;                                                           0811
       IF f2 THEN *cadstr* ← *cadstr*, "    Host: ", ptr3 ptr4,
       EOL;                                                           0812
       END;                                                           0813
    f2 ← getihost(entrystr, 0, $ptr3, $ptr4);                         0814
    IF (f1 ← getinma(entrystr, 0, $ptr1, $ptr2)) OR f2 THEN           0815
       BEGIN                                                          0816
       *cadstr* ← *cadstr*, "  Network", EOL;                         0817
       IF f1 THEN *cadstr* ← *cadstr*, "    User: ", ptr1 ptr2,
       EOL;                                                           0818
       IF f2 THEN *cadstr* ← *cadstr*, "    Host: ", ptr3 ptr4,
       EOL;                                                           0819
       END;                                                           0820
getiadd(entrystr, 0, $ptr1, $ptr2);                                   0821
IF cadstr.L > 0 OR ptr2[1] > ptr1[1] THEN                             0822
    BEGIN                                                             0823
    *statstr* ← *statstr*, "Mail Addresses: ", EOL;                   0824
    IF ptr2[1] > ptr1[1] THEN                                         0825
       *statstr* ← *statstr*, "  Hardcopy Address: ", ptr1 ptr2,
       EOL;                                                           0826
    *statstr* ← *statstr*, *cadstr*;                                  0827
    END;                                                              0828
getiverify(entrystr, 0, $ptr1, $ptr2);                                0829
*statstr* ← *statstr*, ptr1 ptr2, EOL;                                0830
IF getiphone(entrystr, 0, $ptr1, $ptr2) THEN                          0831
    BEGIN                                                             0832
```

```
        *statstr* ← *statstr*, "Phone: ", ptr1 ptr2, EOL;                0833
        END;                                                             0834
    IF getifunction(entrystr, 0, $ptr1, $ptr2) THEN                      0835
        BEGIN                                                            0836
        *statstr* ← *statstr*, "Function: ", ptr1 ptr2, EOL;             0837
        END;                                                             0838
    IF identwheel AND geticapability(entrystr, 0, $ptr1, $ptr2) THEN
                                                                         0839
        BEGIN                                                            0840
        *statstr* ← *statstr*, "Capabilities: ", ptr1 ptr2, EOL;         0841
        END;                                                             0842
    IF getisubcol(entrystr, 0, $ptr1, $ptr2) THEN                        0843
        BEGIN                                                            0844
        *statstr* ← *statstr*, "Sub-Collection: ", ptr1 ptr2, EOL;       0845
        END;                                                             0846
    IF getidelivery(entrystr, 0, $ptr1, $ptr2) THEN                      0847
        BEGIN                                                            0848
        *statstr* ← *statstr*, "Delivery: ", ptr1 ptr2, EOL;             0849
        END;                                                             0850
    IF getimcmnts(entrystr, 0, $ptr1, $ptr2) THEN                        0851
        BEGIN                                                            0852
        *statstr* ← *statstr*, "Comments: ", ptr1 ptr2, EOL;             0853
        END;                                                             0854
    RETURN;                                                              0855
    END.
                                                                         0856
                                                                         0857
FINISH of psjournal                                                      0254
```

PSSYNGEN

(MLK)PSSYNGEN (MLK)PSSYNGEN (MLK)PSSYNGEN (MLK)PSSYNGEN (MLK)PSSY
(MLK)PSSYNGEN (MLK)PSSYNGEN (MLK)PSSYNGEN (MLK)PSSYNGEN (MLK)PSSY

```
< NLS, PSSYNGEN.NLS;2, >, 3-OCT-74 10:59 KEV ;;;;
FILE pssyngen % LIO <rel-nls>pssyngen %% (lio,)
(rel-nls,pssyngen.rel,)%                                                02
% x - level routines for syntax generating subsystem %                 03
(xkshow) PROCEDURE                                                      019
   % FORMALS %                                                          020
      (resultptr, parsemode, type, ent, qual);                          021
   LOCAL adr;                                                           022
   REF resultptr, type, ent, qual;                                      023
   csstkx ← csstkb ← -1;                                               0171
   CASE parsemode OF                                                    024
      = parsing:                                                        025
         BEGIN                                                          026
         CASE qual OF                                                   027
            = $all:  cmdctl ← 0;                                        028
            = $dnls:  cmdctl ← 1;                                       029
            = $tnls:  cmdctl ← 2;                                       030
            ENDCASE cmdctl ← 0;                                         031
         CASE type OF                                                   032
            = $command:                                                 033
               BEGIN                                                    034
               fbctl( typenulllit );                                    035
               ckshwcmd( ent );                                         036
               fbctl( addcalit );                                       037
               END;                                                     038
            = $rule:                                                    039
               BEGIN                                                    040
               IF NOT (adr ← gtadrs( &ent ) ) THEN                      041
                  err($"Invalid command name");                         042
               fbctl( typenulllit );                                    043
               ckshwrul( adr );                                         044
               fbctl( addcalit );                                       045
               END;                                                     046
            = $subsystem:                                               047
               BEGIN                                                    048
               fbctl( typenulllit );                                    049
               ckshwsub( ent );                                         050
               fbctl( addcalit );                                       051
               END;                                                     052
            ENDCASE;                                                    053
         END;                                                           054
      ENDCASE;                                                          055
   RETURN( &resultptr );                                                056
   END.
                                                                        057
   %%                                                                  058
```

```
(xkcopy) PROCEDURE                                                      059
   % FORMALS %                                                          060
      (resultptr, parsemode, type, ent, qual, dest, lvl);               061
   LOCAL adr;                                                           062
   REF resultptr, type, ent, qual, dest, lvl;                           063
   csstkx ← csstkb ← -1;                                               0172
   CASE parsemode OF                                                    064
      = parsing:                                                        065
         BEGIN                                                          066
         CASE qual OF                                                   067
            = $all:  cmdctl ← 0;                                        068
            = $dnls:  cmdctl ← 1;                                       069
            = $tnls:  cmdctl ← 2;                                       070
            ENDCASE cmdctl ← 0;                                         071
         CASE type OF                                                   072
            = $command:                                                 073
               BEGIN                                                    074
               curmkr ← ckcopcmd( dest, lvl, ent );                     075
               curmkr[1] ← 1;                                           076
               END;                                                     077
            = $rule:                                                    078
               BEGIN                                                    079
               IF NOT (adr ← gtadrs( &ent ) ) THEN                      080
                  err($"Invalid command name");                         081
               curmkr ← ckcoprul( dest, lvl, adr );                     082
               curmkr[1] ← 1;                                           083
               END;                                                     084
            = $subsystem:                                               085
               BEGIN                                                    086
               curmkr ← ckcopsub( dest, lvl, ent );                     087
               curmkr[1] ← 1;                                           088
               END;                                                     089
            ENDCASE;                                                    090
         dpset(dspstrc, curmkr, endfil, gethxt(curmkr));                091
         END;                                                           092
      ENDCASE;                                                          093
   RETURN( &resultptr );                                                094
   END.
                                                                        095
   %%                                                                  096
```

```
(xkbuild) PROCEDURE                                                    04
   % FORMALS %                                                         05
      (resultptr, parsemode, asub);                                    06
   REF resultptr, asub;                                                07
   CASE parsemode OF                                                   08
      = parsing:                                                       09
         BEGIN                                                         010
         IF NOT asub THEN                                              011
            asub ← [$sbstack+sbstkx-$sbentsize].sbptr;                 012
         bldsynsub( asub );                                            013
         END;                                                          014
      ENDCASE;                                                         015
   RETURN( &resultptr );                                               016
   END.
                                                                       017
   %%                                                                  018
```

```
(gtadrs) PROCEDURE                                                      097
   % FORMALS %                                                          098
      (ptr);                                                            099
   LOCAL top, bot, i, symval, bp;                                       0100
   LOCAL TEXT POINTER tp1;                                              0101
   LOCAL STRING locstr[100];                                            0102
   REF ptr;                                                             0103
   tp1 ← [&ptr+2];  tp1[1] ← [&ptr+3];                                  0104
   *locstr* ← + ptr tp1;                                                0105
   % convert to rad50 %                                                 0106
      symval ← 0;                                                       0107
      bp ← 440700000001B + $locstr;                                     0108
      FOR i ← 1 UP UNTIL > MIN(6,locstr.L) DO                           0109
         symval ← symval * 50B + ↑bp - 54;                              0110
   % get limits of symbol table %                                       0111
      bot ← 116B;  bot ← [bot];                                         0112
      top ← bot.LH;  IF bot < 0 THEN top.LH ← -1;                       0113
      bot ← bot.RH;  top ← bot - tcp - 2;                               0114
   % now look for symbol %                                             0115
      FOR top DOWN 2 UNTIL < bot DO                                     0116
         IF ( [top] .A 32M ) = symval THEN                              0117
            RETURN( [top+1] );                                          0118
   RETURN( FALSE );                                                     0119
   END.
                                                                        0120
   %%                                                                   0121
```

```
(bldsynsub) PROCEDURE( asub );                                          0122
   LOCAL tmp2, cptr, instptr, tptr, i;                                  0123
   REF asupsub, acursub, cptr, instptr, tptr;                           0124
   tmp2 ← getsdptr($usys1,$allsubs);                                    0125
   IF (asub = (&acursub := asub)) AND                                   0126
      (tmp2 = (&asupsub := tmp2)) THEN RETURN;                          0127
   % build store node %                                                 0128
      synstore.alternative ←                                            0129
         synstore.nsuccessor ← synstore.val2 ← synstore.ctrl ← 0;       0130
      synstore.opcode ← $store;                                         0131
      synstore.addr ← $syncur;                                          0132
   &cptr ← &tptr ← $synsubs;                                            0133
   FOR i ← 0 UP UNTIL > 1 DO                                            0134
      BEGIN                                                             0135
      CASE i OF                                                         0136
         = 0: &instptr ← acursub.dptrun;                                0137
         = 1:                                                           0138
            IF acursub = asupsub THEN EXIT LOOP                         0139
            ELSE &instptr ← asupsub.dptrun;                             0140
         ENDCASE;                                                       0141
      WHILE &instptr AND (&cptr < $synstore) DO                         0142
         IF instptr.opcode # $keyop THEN                                0143
            &instptr ← instptr.alternative                              0144
         ELSE                                                           0145
            BEGIN                                                       0146
            % build keyword node %                                      0147
               cptr.opcode ← $keyop;                                    0148
               cptr.ctrl ← instptr.ctrl;                                0149
               cptr.val2 ← instptr.val2;                                0150
               cptr.addr ← instptr.addr;                                0151
               cptr.alternative ← 0;                                    0152
               IF &tptr NOT= &cptr THEN                                 0153
                  BEGIN                                                 0154
                  tptr.alternative ← &cptr;                             0155
                  &tptr ← &cptr;                                        0156
                  END;                                                  0157
               &cptr ← cptr.nsuccessor ← &cptr+2;                       0158
            % build enter node %                                        0159
               cptr.alternative ← cptr.val2 ← cptr.ctrl ← 0;            0160
               cptr.opcode ← $enter;                                    0161
               cptr.addr ← &instptr;                                    0162
               cptr.nsuccessor ← $synstore;                             0163
            &cptr ← &cptr+2;                                            0164
            &instptr ← instptr.alternative;                             0165
            END;                                                        0166
      END;                                                              0167
   RETURN;                                                              0168
   END.
                                                                        0169
FINISH                                                                  0170
```

PS SYSTEM

(MLK)PSSYSTEM (MLK)PSSYSTEM (MLK)PSSYSTEM (MLK)PSSYSTEM (MLK)PSSY
(MLK)PSSYSTEM (MLK)PSSYSTEM (MLK)PSSYSTEM (MLK)PSSYSTEM (MLK)PSSY

```
< NLS, PSSYSTEM.NLS;22, >, 4-OCT-74 10:51 DSM ;;;;
FILE pssystem % LLO <rel-nls>pssystem %% (LLO,) (rel-nls,pssystem.rel,)
%                                                                    02
% SUPERVISOR SUBSYSTEM %                                             09
   (xgoto) PROC( % GOTO execution routine %                          010
      % FORMAL ARGUMENTS %                                           011
         resultptr,     % ptr to result record %                     012
         parsemode,     % parsing mode %                             013
         subsysptr,     % ptr to subsys name record %                014
         dispatchptr,   % ptr to subsystem dispatch pointer %       015
         exflagptr);    % ptr to EXECUTE flag record %               016
      % RETURNS %                                                    017
         % 1) resultptr for TRUE return %                            018
      % ABNORMAL RETURNS %                                           019
         % call to err if sbstack overflows %                        020
      LOCAL % VARIABLES %                                           021
         entry,% ptr to new subsystem stack entry %                 022
         count,% command repetition count %                         023
         nptr,          % ptr to subsystem name %                    024
         gptr;          % ptr to location in grammar %               025
      REF % VARIABLES %                                             026
         entry,                                                     027
         dispatchptr,                                               028
         resultptr,                                                 029
         subsysptr,                                                 030
         exflagptr,                                                 031
         nptr,                                                      032
         gptr;                                                      033
      %-------------%                                               034
      CASE parsemode OF                                             035
         = parsing:                                                 036
            BEGIN                                                   037
            IF NOT exflagptr AND subsysptr = $tenex THEN xgoexec()  038
            ELSE                                                    039
               BEGIN                                                040
               count ← IF exflagptr THEN 1 ELSE -1;                 041
               &nptr ← subsysptr;                                   042
               % set the grammar address according to which subsystem
               we're going to %                                     043
                  &gptr ← dispatchptr;                              044
               % build a new subsystem stack entry if there is room %
                                                                    045
                  IF sbstkx < $sbstksize                            046
                     THEN                                           047
                        BEGIN                                       048
                        % save the current subsystem mode and set it to
                        rentry %                                    049
                           &entry ← $sbstack + sbstkx - $sbentsize; 050
                           entry.sbpmode ← entry.sbmode;            051
                           entry.sbmode ← sbrentry;                 052
                        % set up a new substack entry %            053
                           &entry ← &entry + $sbentsize;            054
                           entry.sbmode ← sbstart;                  055
                           entry.sbptr ← &gptr;                     056
                           entry.sbnptr ← &nptr;                    057
                           entry.sbcount ← count;                   058
```

```
                        sbstkx ← sbstkx + $sbentsize;                    059
                    END                                                  060
                ELSE err( $"subsystem stack overflow: Please quit
                out of at least one subsystem");                         061
            END;                                                         062
        END;                                                             063
    ENDCASE;                                                             064
    RETURN (&resultptr);                                                 065
    END.
                                                                         066
(xquit) PROC( % QUIT execution routine %                                 067
    % FORMAL ARGUMENTS %                                                 068
      resultptr,     % ptr to result record %                           069
      parsemode,     % parsing mode %                                   070
      subsysptr);    % ptr to subsys name record %                      071
    % RETURNS %                                                         072
      % 1) resultptr for TRUE return %                                  073
    % ABNORMAL RETURNS %                                                074
      % call to err if subsystem is not in subsys stack %               075
    LOCAL % VARIABLES %                                                  076
      level,         % level in sbstack to cut back to %                077
      index,         % loop var, temp level counter %                   078
      entry,% ptr to new subsystem stack entry %                         079
      nptr,          % ptr to subsystem name %                          080
      gptr;          % ptr to location in grammar %                     081
    LOCAL STRING                                                         082
      errmess[50];  % error diagnostic string %                         083
    REF % VARIABLES %                                                    084
      entry,                                                             085
      resultptr,                                                         086
      subsysptr,                                                         087
      exflagptr,                                                         088
      nptr,                                                              089
      gptr;                                                              090
    %-------------%                                                      091
    CASE parsemode OF                                                    092
      = parsing:                                                         093
        BEGIN                                                            094
        % reset the cueflag so prompts will come out properly %          095
          cueflg ← FALSE;                                                096
        dpset(dspno, endfil, endfil, endfil);                            097
        % set level to mark how far the subsystem stack is to be
        popped, level is set to the anticipated value for sbstkx
        after we've quit out of 1/more subsytems  %                      098
          CASE (&nptr ← subsysptr) OF                                    099
            = 0:          % default, exit current subsystem only %
                                                                         0100
              level ← sbstkx - $sbentsize; % cut top entry only
              %                                                          0101
            = $nls:       % quit NLS altogether %                       0102
              level ← 0;                                                 0103
          ENDCASE                                                        0104
            BEGIN % search back through sbstack for match with
            given name %                                                 0105
            FOR level ← sbstkx-$sbentsize DOWN $sbentsize UNTIL <
            0 DO                                                         0106
```

```
                BEGIN                                                   0107
                &entry ← $sbstack + level;                              0108
                IF entry.sbnptr = &nptr THEN                            0109
                   BEGIN                                                0110
                   level ← level + $sbentsize;                          0111
                   EXIT CASE;                                           0112
                   END;                                                 0113
                END;                                                    0114
             % no match was found, so put out an error %                0115
                *errmess* ← *nptr*, " is not in your subsystem
                stack";                                                 0116
                err( $errmess );                                        0117
             END;                                                       0118
          % proceeed down the subsystem stack marking subsystems to
          be exited %                                                   0119
             FOR index ← sbstkx-$sbentsize DOWN $sbentsize UNTIL <
             level DO                                                   0120
                BEGIN                                                   0121
                &entry ← $sbstack + index;                              0122
                entry.sbmode ← scfinish; % set to exit from subsystem
                %                                                       0123
                END;                                                    0124
          END;                                                          0125
       ENDCASE;                                                         0126
       RETURN (&resultptr);                                             0127
       END.
                                                                        0128
(xksyntax) PROCEDURE % show syntax of a command %                       0913
       % FORMALS %                                                      0914
          (resultptr, parsemode, anode);                                0915
       LOCAL adr;                                                       0916
       REF resultptr, anode;                                            0917
       csstkx ← csstkb ← -1;                                            0930
       CASE parsemode OF                                                0918
          = parsing:                                                    0919
            BEGIN                                                       0920
            cmdctl ← IF nlmode = fulldisplay THEN 1 ELSE 2;             0921
            fbctl( typenulllit );                                       0922
            ckshwcmd( anode );                                          0923
            fbctl( addcalit );                                          0924
            END;                                                        0925
          ENDCASE;                                                      0926
       RETURN( &resultptr );                                            0927
       END.
                                                                        0928
       %%                                                               0929
```

```
FE (xsublist) PROC( % show current subsystem stack %                    0129
    % FORMAL ARGUMENTS %                                                0130
       resultptr,     % ptr to result record %                          0131
       parsemode);    % parsing mode %                                  0132
    LOCAL % variables %                                                 0133
       entryptr;      % ptr to substack entry %                         0134
    LOCAL STRING str[300];                                              0135
    REF                                                                 0136
       entryptr,                                                        0137
       resultptr;                                                       0138
    %----------%                                                        0139
    CASE parsemode OF                                                   0140
       = parsing:                                                       0141
          BEGIN                                                         0142
          % initialize display string %                                 0143
             *str* ← "*** Subsystem Stack: (current one first) ***
             ";                                                         0144
          % append subsystem names to str %                            0145
             FOR &entryptr ← $sbstack+sbstkx-$sbentsize DOWN
             $sbentsize UNTIL < $sbstack DO                             0146
                *str* ← *str*, *[entryptr.sbnptr]*, EOL;                0147
          % put a trailer on the string %                              0148
             *str* ← *str*, "***";                                      0149
          % output the string %                                        0150
             fbctl( typecalit, $str );                                  0151
          END;                                                          0152
    ENDCASE;                                                            0153
    RETURN (&resultptr);                                                0154
    END.                                                                0155

FE (xsubcurrent) PROC( % show current subsystem %                       0156
    % FORMAL ARGUMENTS %                                                0157
       resultptr,     % ptr to result record %                          0158
       parsemode);    % parsing mode %                                  0159
    LOCAL STRING str[40];                                               0162
    REF                                                                 0163
       resultptr;                                                       0165
    %----------%                                                        0166
    CASE parsemode OF                                                   0167
       = parsing:                                                       0168
          BEGIN                                                         0169
          % fetch subsystem name to str %                              0170
             xgtcrsb( $str );                                           0539
          % output the string %                                        0173
             fbctl( typecalit, $str );                                  0174
          END;                                                          0175
    ENDCASE;                                                            0176
    RETURN (&resultptr);                                                0177
    END.                                                                0178

FE (xgtcrsb) PROC( % get current subsystem name to string %             0516
    % FORMAL ARGUMENTS %                                                0517
       astr);         % adr string to get current name %                0518
    LOCAL % variables %                                                 0520
       entryptr;      % ptr to substack entry %                         0521
    REF                                                                 0523
```

```
    entryptr,                                                          0524
    astr;                                                              0525
  %----------%                                                         0526
  % fetch subsystem name to str %                                      0530
    &entryptr ← $sbstack+sbstkx-$sbentsize;                            0531
    *astr* ← *[entryptr.sbnptr]*;                                      0532
  RETURN;                                                              0537
  END.                                                                 0538

(abortsubsystem) PROC( % aborts execution of a subsystem, prior to
normal termination.  Accomplishes what would be accomplished as if
the user had typed in QUIT CA %                                        0179
  % FORMAL ARGUMENTS %                                                 0180
    errstrptr);   % ptr to diagnostic string %                        0181
  LOCAL % variables %                                                  0182
    entry;        % ptr to subsystem stack entry %                    0183
  REF errstrptr, entry;                                                0184
  % ---------------- %                                                0185
  % change the mode of the current subsystem stack entry to exit
  the subsystem %                                                      0186
    &entry ← $sbstack + sbstkx - $sbentsize;                           0187
    entry.sbmode ← sbfinish;                                           0188
  % display the diagnostic string, generatiing a SIGNAL which
  causes us to exit the current subsystem after involing the
  termination rule (if any ) %                                         0189
    err( &errstrptr );                                                 0190
  END.                                                                 0191

%jump%                                                                 0711
  (xjump) %Execute Jump Command%                                       0712
    PROCEDURE                                                          0713
      %FORMALS%                                                        0714
        (result,          %result record%                            0715
        parsemode,        %parsing, backup, cleanup%                 0716
        entity,           %type of load%                             0717
        destination,      %dest record%                              0718
        vs);              %view specs%                               0719
        REF                                                            0720
          result, destination, vs, entity;                             0721
      LOCAL da, start, oldda, fileno, linkp, vsupdate, adstr[40];      0722
      REF da, oldda, linkp;                                            0723
      LOCAL TEXT POINTER t1, t2, csp;                                  0724
      LOCAL STRING locstr[200];                                        0725
      %----------------------%                                         0726
      CASE parsemode OF                                                0727
        = parsing:                                                     0728
          BEGIN                                                        0729
          &da ← cspupdate ← lda();                                     0730
          csp ← da.dacsp;  csp[1] ← da.daccnt;                         0731
          cspvs ← da.davspec;                                          0732
          cspvs[1] ← da.davspc2;                                       0733
          cspcacode ← da.dacacode;                                     0734
          cspusqcod ← da.dausqcod;                                     0735
          vsupdate ← TRUE;                                             0736
          start ← &destination;                                        0737
```

```
CASE entity OF                                                          0738
   = $item:                                                             0739
     BEGIN                                                              0740
     (xj0);                                                             0741
     IF cspupdate THEN                                                  0742
        BEGIN                                                           0743
        curmkr ← destination;                                           0744
        curmkr[1] ← IF nlmode = fulldisplay THEN 1 ELSE
        destination[1];                                                 0745
        END;                                                            0746
     IF vsupdate THEN                                                   0747
        BEGIN                                                           0748
        cspvs ← vs;                                                     0749
        cspvs[1] ← vs[1];                                               0750
        cspcacode ← vs.vscacode;                                        0751
        cspusqcod ← vs.vsusqcod;                                        0752
        END;                                                            0753
     dpset(dspjpf, curmkr, endfil, endfil);                             0754
     RETURN(&result);                                                   0755
     END;                                                               0756
   = $itemnovs:                                                         0931
     BEGIN                                                              0932
     vsupdate ← FALSE;                                                  0933
     GOTO xj0;                                                          0934
     END;                                                               0935
   = $successor:        *locstr* ← ".s";                                0757
   = $predecessor:      *locstr* ← ".p";                                0758
   = $up:               *locstr* ← ".u";                                0759
   = $down:             *locstr* ← ".d";                                0760
   = $head:             *locstr* ← ".h";                                0761
   = $tail:             *locstr* ← ".t";                                0762
   = $end:              *locstr* ← ".e";                                0763
   = $back:             *locstr* ← ".b";                                0764
   = $next:             *locstr* ← ".n";                                0765
   = $origin:           *locstr* ← ".o";                                0766
   = $filereturn:                                                       0767
     BEGIN                                                              0768
     curmkr ← destination.retstid;                                      0769
     FIND SF(*[destination.retfn]*) ↑tl;                                0770
     CASE lnbfls( $tl, 0, $locstr) OF                                   0771
        = lhostn: NULL;                                                 0772
        ENDCASE                                                         0773
           err($"Remote File Manipulations Not
           Implemented Yet");                                           0774
     curmkr.stfile ← cloafil($locstr);                                  0775
     curmkr[1] ← destination.retcc;                                     0776
     vs ← destination.retvs1;                                           0777
     vs[1] ← destination.retvs2;                                        0778
     vs.vscacode ← cspcacode;                                           0779
     vs.vsusqcod ← cspusqcod;                                           0780
     usesrr ← destination.retsrr;                                       0781
     destination ← curmkr;                                              0782
     destination[1] ← curmkr[1];                                        0783
     GOTO xj0;                                                          0784
     END;                                                               0785
   = $return:                                                           0786
```

```
                BEGIN                                                   0787
                curmkr ← destination.retstid;                           0788
                curmkr[1] ← destination.retcc;                          0789
                vs ← destination.retvs1;                                0790
                vs[1] ← destination.retvs2;                             0791
                vs.vscacode ← cspcacode;                                0792
                vs.vsusqcod ← cspusqcod;                                0793
                destination ← curmkr;                                   0794
                destination[1] ← curmkr[1];                             0795
                GOTO xj0;                                               0796
                END;                                                    0797
            = $link:                                                    0798
                BEGIN                                                   0799
                vsupdate ← FALSE;                                       0800
                IF destination.stastr THEN                              0801
                   BEGIN                                                0802
                   start ← $csp;                                        0803
                   t1 ← destination; t1[1] ← destination[1];            0804
                   t2 ← [&destination+d2sel];                           0805
                      t2[1] ← [&destination+d2sel+1];                   0806
                   END                                                  0807
                ELSE                                                    0808
                   BEGIN                                                0809
                   lnkprs( &destination, $adstr);                       0810
                   t1 ← adstr[ls]; t1[1] ← adstr[ls+1];                 0811
                   t2 ← adstr[le]; t2[1] ← adstr[le+1];                 0812
                   END;                                                 0813
                GOTO xj2;                                               0814
                END;                                                    0815
            = $name:                                                    0816
                BEGIN                                                   0817
                t1 ← destination;                                       0818
                t1[1] ← [&destination+d2sel+1];                         0819
                *locstr* ← destination t1;                              0820
                start ← $csp;                                           0821
                END;                                                    0822
            = $filename:                                                0906
                BEGIN                                                   0907
                t1 ← destination;                                       0908
                t1[1] ← [&destination+d2sel+1];                         0909
                *locstr* ← '(, destination t1, ",)";                    0910
                start ← $csp;                                           0911
                END;                                                    0912
            = $file:                                                    0823
                BEGIN                                                   0824
                lnkprs(&destination, $adstr);                           0825
                t1 ← adstr[ls];  t1[1] ← adstr[ls+1];                   0826
                t2 ← adstr[fe];  t2[1] ← adstr[fe+1];                   0827
                *locstr* ← t1 t2, ",)";                                 0828
                start ← $csp;                                           0829
                END;                                                    0830
            = $firstname:                                               0831
                BEGIN                                                   0832
                t1 ← destination;                                       0833
                t1[1] ← [&destination+d2sel+1];                         0834
                *locstr* ← ".c*", destination t1;                       0835
```

```
            start ← $csp;                                                0836
            END;                                                         0837
         = $nextname:                                                    0838
            BEGIN                                                        0839
            t1 ← destination;                                            0840
            t1[1] ← [&destination+d2sel+1];                              0841
            *locstr* ← '*, destination t1;                               0842
            start ← $csp;                                                0843
            END;                                                         0844
         = $extname:                                                     084
            BEGIN                                                        0846
            t1 ← destination;                                            0847
            t1[1] ← [&destination+d2sel+1];                              0848
            *locstr* ← '&, destination t1;                               0849
            start ← $csp;                                                0850
            END;                                                         0851
         = $firstcontent:                                                0852
            BEGIN                                                        0853
            t1 ← destination;                                            0854
            t1[1] ← [&destination+d2sel+1];                              0855
            *locstr* ← ".c", '", destination t1, '", "=C";               0856
            start ← $csp;                                                0857
            END;                                                         0858
         = $nextcontent:                                                 0859
            BEGIN                                                        0860
            t1 ← destination;                                            0861
            t1[ 1] ← [&destination+d2sel+1];                              0862
            *locstr* ← ".n", '", destination t1, '", "=C";               0863
            start ← $csp;                                                0864
            END;                                                         0865
         = $firstword:                                                   0866
            BEGIN                                                        0867
            t1 ← destination;                                            0868
            t1[1] ← [&destination+d2sel+1];                              0869
            *locstr* ← ".c", '", destination t1, '", "=W";               0870
            start ← $csp;                                                0871
            END;                                                         0872
         = $nextword:                                                    0873
            BEGIN                                                        0874
            t1 ← destination;                                            0875
            t1[1] ← [&destination+d2sel+1];                              0876
            *locstr* ← ".n", '", destination t1, '", "=W";               0877
            start ← $csp;                                                0878
            END;                                                         0879
      ENDCASE err(notyet);                                               0880
   (xj1): %set up text pointers and evaluate address
   expression%                                                           0881
   FIND SF(*locstr*) ↑t1 SE(*locstr*) ↑t2;                               0882
   (xj2): %assuming the text pointers are set up, evaluate
   address expression%                                                   0883
   IF vsupdate THEN                                                      0884
      BEGIN                                                              0885
      da.davspec ← vs;                                                   0886
      da.davspc2 ← vs[1];                                                0887
      da.dacacode ← vs.vscacode;                                         0888
      da.dausqcod ← vs.vsusqcod;                                         0889
```

```
            END;                                                    0890
         vs ← caddexp($t1, $t2, &da, start : vs[1], vs.vscacode,
         vs.vsusqcod, usesrr);                                      0891
         IF vsupdate THEN                                           0892
            BEGIN                                                   0893
            da.davspec ← cspvs;                                     0894
            da.davspc2 ← cspvs[1];                                  0895
            da.dacacode ← cspcacode;                                0896
            da.dausqcod ← cspusqcod;                                0897
            END;                                                    0898
         vsupdate ← TRUE;                                           0899
         destination ← destination[d2sel] ← [start];                0900
            destination[1] ← destination[d2sel+1] ← [start+1];
                                                                    0901
         GOTO xj0;                                                  0902
         END;                                                       0903
      ENDCASE RETURN(&result);                                      0904
   END.
                                                                    0905
% TENEX SUBSYSTEM %                                                0192
   (xgoExec) PROCEDURE;                    %goto exec command%     0193
      <AUXCOD, gofork>($"<SYSTEM>EXEC.SAV", -1, -1, 4B10);          0194
      dpset(dspallf, endfil, endfil, endfil);                       0195
      RETURN END.
                                                                    0196
% UTILITY SUBSYSTEM %                                              0197
   % this file contains the "x" routines for support of the LIBENT
   parser%                                                          0198
   (xutilinit) PROCEDURE( %initialize stuff for utility functions%  0361
      % FORMAL ARGUMENTS %                                         0362
      resultptr,          %ptr to result record%                    0363
      pmode);             %parsing mode%                            0364
      REF resultptr;                                                0365
      LOCAL STRING uname[10];                                       0366
      %--------------%                                             0367
      CASE pmode OF                                                 0368
         = parsing:                                                 0369
            BEGIN                                                   0370
            *uname* ← "UTILTY";                                     0371
            nlssbn ← getsbn($uname); %convert to sixbit%            0372
            !setnm(nlssbn);                                         0373
            libflg ← oldflg ← liblod ← jdfi ← jdid ← jdno ← 0;
            %library functions, parameter strings null%             0374
            END;                                                    0375
      ENDCASE;                                                      0376
      RETURN( &resultptr);                                          0377
      END.
                                                                    0378
   (xjoutil) PROCEDURE( %run the journal - maintenance mode%        0313
      % FORMAL ARGUMENTS %                                         0314
      resultptr,          %ptr to result record%                    0315
      pmode);  %parsing mode%                                       0316
      REF resultptr;                                                0321
      %--------------%                                             0322
      CASE pmode OF                                                 0323
         = parsing:                                                 0324
```

```
        BEGIN                                                          0325
        jnlin(); % roll in JNLDEL file and start processing. %         0409
        END;                                                           0350
     = cleanup:                                                        0351
        BEGIN                                                          0352
        IF jdid THEN freestring(jdid := 0, $dspblk);                   0464
        IF jdfi THEN freestring(jdfi := 0, $dspblk);                   0465
        IF jdno THEN freestring(jdno := 0, $dspblk);                   0466
        jnlout(); % reset buffer-- roll out JNLDEL %                   0411
        END;                                                           0357
     ENDCASE;                                                          0358
  RETURN( &resultptr);                                                 0359
  END.                                                                 0360

(jnlout)PROCEDURE; % Roll out JNLDEL file %                            0406
  IF jnlprog THEN % Reset program buffer. %                            0353
     BEGIN                                                             0354
     gpgmrst(); % Reset stack %                                        0355
     END;                                                              0356
  RETURN;                                                              0412
  END.                                                                 0407

(xcutil) PROCEDURE(  %run tasks%                                       0232
  % FORMAL ARGUMENTS %                                                0233
  resultptr,  %ptr to result record%                                   0234
  pmode,   %parsing mode%                                              0235
  umode);   %flags to be passed to nlsutility%                         0236
  REF resultptr, umode;                                                0237
  %simply call utility function with prior specified mode%             0238
     CASE pmode OF                                                     0239
        = parsing:                                                     0240
           BEGIN                                                       0241
           typeas($"
             *** Running Tasks ***");                                  0242
            nlsutility(CASE umode OF                                   0243
              = $detached: 4;                                          0310
              = $file: 5;                                              0309
              = $tty: 6;                                               0311
              ENDCASE 5);                                              0312
           END;                                                        0244
     ENDCASE;                                                          0245
     RETURN( &resultptr);                                              0246
     END.                                                              0247
(xsetjou) PROCEDURE(  %set flags for journal run%                      0248
  % FORMAL ARGUMENTS %                                                0249
  resultptr,          %ptr to result record%                           0250
  pmode,              %parsing mode%                                   0251
  option);                  %pointer to option specified%              0252
  REF resultptr, option;                                               0253
  CASE pmode OF                                                        0254
     = parsing:                                                        0255
        BEGIN                                                          0256
        CASE option OF                                                 0257
           = $auto:          libflg ←libflg .V 20B;                    0258
           = $continue:      libflg ←libflg .V 10B;                    0259
           = $on:            libflg ←libflg .V 2B;                     0260
```

```
            = $recover:     libflg ←libflg .V 1B;                    0262
            = $slinker:     libflg ←libflg .V 4B;                    0263
            = $update:      libflg ←libflg .V 40B;                   0264
            ENDCASE  typeas($"invalid option specified");            0265
         END;                                                        0266
   ENDCASE;                                                          0267
   RETURN( &resultptr);                                              0268
   END.                                                              0269

(xsetpar) PROCEDURE(  %set flags for partial journal delivery%       0467
   % FORMAL ARGUMENTS %                                              0468
   resultptr,        %ptr to result record%                          0469
   pmode,            %parsing mode%                                  0470
   entity,           %pointer to entity selected%                    0471
   option);                %pointer to option specified%             0472
   LOCAL TEXT POINTER z1, z2;                                        0515
   REF resultptr, entity, option;                                    0473
   CASE pmode OF                                                     0474
      = parsing:                                                     0475
         BEGIN                                                       0476
         IF option # $clear THEN                                     0477
            BEGIN                                                    0478
            z1 ← entity;                                             0479
            z1[1] ← [&entity + 1];                                   0480
            z2 ←  [&entity  + 2];                                    0481
            z2[1] ← [&entity + 3];                                   0482
            END;                                                     0483
         CASE option OF                                              0484
            = $clear:                                                0485
               BEGIN                                                 0486
               oldflg ← 0;                                           0487
               IF jdid THEN freestring(jdid := 0, $dspblk);          0488
               IF jdfi THEN freestring(jdfi := 0, $dspblk);          0489
               IF jdno THEN freestring(jdno := 0, $dspblk);          0490
               END;                                                  0491
            = $idents:                                               0492
               BEGIN                                                 0493
               jdid ← getstring(1000, $dspblk);                      0494
               *[jdid]* ← z1 z2;                                     0495
               oldflg ← oldflg .V 200B;                              0496
               END;                                                  0497
            = $files:                                                0498
               BEGIN                                                 0499
               jdfi ← getstring(1000, $dspblk);                      0500
               *[jdfi]* ← z1 z2;                                     0501
               oldflg ← oldflg .V 100B;                              0502
               END;                                                  0503
            = $number:                                               0504
               BEGIN                                                 0505
               jdno ← getstring(1000, $dspblk);                      0506
               *[jdno]* ← z1 z2;                                     0507
               oldflg ←oldflg .V 400B;                               0508
               END;                                                  0509
            ENDCASE  dismes(2,$"invalid option specified");          0510
         END;                                                        0511
   ENDCASE;                                                          0512
```

```
   RETURN ( &resultptr);                                                0513
   END.
                                                                        0514
(xsetld) PROCEDURE(  %set load average cutoff value%                    0379
   % FORMAL ARGUMENTS %                                                 0380
   resultptr,        %ptr to result record%                             0381
   pmode,            %parsing mode%                                     0382
   param);           %pointer to selection record%                      0383
   LOCAL mlav[2], tp;                                                   0384
   LOCAL STRING lavstr[30];                                             0385
   REF resultptr, param, tp;                                            0386
   CASE pmode OF                                                        0387
      = parsing:                                                        0388
         BEGIN                                                          0389
         &tp ← &param + d2sel;                                          0390
         *lavstr* ← param tp;                                           0391
         nfloat($lavstr, $mlav, $mlav + 1);                             0392
         oljmlav ← mlav;                                                0393
         liblod ← TRUE; % set flag which says load average has been
         set by hand;  JOUTIL checks it to see if it has to set the
         load average cutoff. %                                         0394
         END;                                                           0395
      ENDCASE;                                                          0396
   RETURN ( &resultptr);                                                0397
   END.
                                                                        0398
(xpriority)          % set priority only if person can enable %         0413
   PROCEDURE (resultptr, parsemode, priorval);                          0414
   LOCAL value, char;                                                   0415
   LOCAL TEXT POINTER z1, z2;                                           0416
   LOCAL STRING pristr[10];                                             0417
   REF resultptr, priorval;                                             0418
   CASE parsemode OF                                                    0419
      = parsing:                                                        0420
         BEGIN                                                          0421
         % get information out of the parameter records %               0422
            z1 ← priorval;                                              0423
            z1[1] ← [&priorval + 1];                                    0424
            z2 ← [&priorval + 2];                                       0425
            z2[1] ← [&priorval + 3];                                    0426
            *pristr* ← z1 z2;                                           0427
            % Remove blanks %                                           0428
            IF NOT FIND SF(*pristr*) $NP ↑z1 $D ↑z2 THEN                0429
               err($"Illegal priority value");                          0430
            *pristr* ← z1 z2;                                           0431
         % check the validity of the value %                           0432
            CASE pristr.L OF                                            0433
               = 0: value ← 202B;                                       0434
               = 1:                                                     0435
                  IF *pristr*[1] = '0 THEN value ← 0                    0436
                  ELSE REPEAT CASE(4); % Force error %                  0437
               = 3:                                                     0438
                  BEGIN                                                 0439
                  IF (char ← *pristr*[1]) IN ['1, '3] THEN              0440
                     value ← (char - '0) * 100B                         0441
                  ELSE REPEAT CASE(4); % Force error %                  0442
```

```
                    IF *pristr*[2] # '0 THEN REPEATCASE(4);                 0443
                    IF (char ← *pristr*[3]) IN ['1, '3] THEN                0444
                       value ← (char - '0) + value                          0445
                    ELSE REPEAT CASE(4); % Force error %                    0446
                    END;                                                    0447
                 ENDCASE err($"Illegal priority value");                    0448
           % set the priority %                                             0449
              setpriority ( value );                                        0450
           END;                                                             0451
   ENDCASE;                                                                 0452
   RETURN( &resultptr );                                                    0453
   END.                                                                     0454

(xxddt)      % go to DDT %                                                  0540
   PROCEDURE (resultptr, parsemode);                                        0541
   REF resultptr;                                                           0545
   CASE parsemode OF                                                        0546
      = parsing:  ddt();                                                    0547
      ENDCASE;                                                              0579
   RETURN( &resultptr );                                                    0580
   END.                                                                     0581

(ddt)  % dummy procedure to get us into ddt %                               0582
   PROCEDURE;                                                               0583
   RETURN;                                                                  0584
   END.                                                                     0585

(xxcheck) PROCEDURE  % Check results of running tasks %                     0586
   (resultptr, parsemode);                                                  0593
   REF resultptr;                                                           0587
   CASE parsemode OF                                                        0588
      = parsing:  inptrf ← tskerrcnt;                                       0589
      ENDCASE;                                                              0590
   RETURN( &resultptr );                                                    0591
   END.                                                                     0592

(xxdetach) PROCEDURE (resultptr, parsemode, infile, outfile);               0594
   LOCAL                                                                    0595
      rhosti, rhosto, inparam, outparam;                                    0596
   LOCAL STRING                                                             0597
      instring[200], outstring[200];                                        0598
   REF resultptr, infile, outfile;                                          0599
   CASE parsemode OF                                                        0600
      = parsing:                                                            0601
         BEGIN                                                              0602
         rhosti ← rhosto ← lhostn;                                          0603
         inparam ← outparam ← FALSE;                                        0604
         IF infile THEN                                                     0605
            BEGIN                                                           0606
            rhosti←inbfls(&infile, 0, $instring);                           0607
            inparam ← $instring;                                            0608
            END;                                                            0609
         IF outfile THEN                                                    0610
            BEGIN                                                           0611
            rhosto←inbfls(&outfile, 0, $outstring);                         0612
            outparam ← $outstring;                                          0613
```

```
            END;                                                    0614
          cxdetach( rhosti, inparam,  rhosto, outparam);            0615
          END;                                                      0619
        ENDCASE;                                                    0616
      RETURN( &resultptr );                                         0617
      END.
                                                                    0618
(xsubnotimp)            % subsystem not yet implemented %           0300
   PROCEDURE (resultptr, parsemode);                                0301
   REF resultptr;                                                   0302
   CASE parsemode OF                                                0303
      = parsing: err(%"subsystem not implemented yet.  Use QUIT command
      to return");                                                  0304
   ENDCASE;                                                         0305
   RETURN( &resultptr );                                            0306
   END.
                                                                    0307
FINISH of pssystem                                                  0308
```

PSUPPORI

(MLK)PSUPPORT (MLK)PSUPPORT (MLK)PSUPPORT (MLK)PSUPPORT (MLK)PSUP
(MLK)PSUPPORT (MLK)PSUPPORT (MLK)PSUPPORT (MLK)PSUPPORT (MLK)PSUP

```
< NLS, PSUPPORT.NLS;82, >, 23-OCT-74 10:46 DSM ;;;;  % PARSER SUPPORT
CODE %
FILE psupport % L10 <rel-nls>psupport %% (L10,) (rel-nls,psupport.rel,)
%                                                                      02
% Declarations %                                                       03
   REGISTER r1=1, r2=2, r3=3, r4=4;                                    04
   REF msgda, rawchr, inpt, tda;                                       05
   DECLARE EXTERNAL outpathstk = 999879;  %value for SIGNAL code%   0811
   DECLARE                                                             06
      nofile = 0, noct = 0,ctcsp = 1, ctfrz = 2,                       07
      ctcpfz = 3, ctmkr = 8, ctcmk = 9, ctcfm = 11, ctlcfm = 15,
      copyflag = 1, moveflag = 2;                                      08
                                                                       09
   DECLARE notyet = 7;                                                010
% PARSING FUNCTIONS %                                                 014
   % READS A SPACE %                                                  015
      (sp) PROC(curptr  parsemode, string);                            016
         % sp looks at the next inpt character, if it is a space, then
         the space is read and a true return is taken.  If the next
         character is not a space, then it is not read, and FALSE is
         returned %                                                    017
         REF curptr, string;                                           018
         %------------%                                               019
         CASE parsemode OF                                             020
            = parsing:                                                 021
               CASE lookc() OF                                         022
                  = SP:                                                023
                     inpt();                                           024
               ENDCASE RETURN (FALSE);                                 025
            = parsehelp:                                               026
               *string* ← "SP:";                                       027
            = parseqmark:                                             0677
               BEGIN                                                  0679
               *string* ← "<SPACE>";                                  0678
               RETURN;                                                0680
               END;                                                   0681
         ENDCASE;                                                      028
         RETURN (&curptr);                                             029
         END.
                                                                       030
                                                                       031
   % READS AN OPTION CHARACTER %                                       032
      (readoption) PROCEDURE( % parsing function which looks at the
      next input char.  If it is an option character, then it reads the
      char and returns TRUE, otherwise it returns FALSE %              033
         % FORMAL ARGUMENTS %                                          034
            resultptr,          % ptr to the result record %          035
            parsemode,          % parsing mode %                      036
            string);   % ptr to help string %                        037
         REF resultptr, string, inpt;                                  038
         % ------------- %                                            039
            CASE parsemode OF                                          040
               = parsing:                                              041
                  CASE lookc() OF                                      042
                     = optchar:                                        043
                        inpt();                                        044
```

```
                ENDCASE RETURN (FALSE);                                 045
            = parsenelp:                                                046
                IF inprompts = partprompts THEN                         0981
                    *string* ← NULL                                     0982
                ELSE                                                    0983
                    *string* ← "OPT:";                                  047
            = parseqmark:                                               0682
                BEGIN                                                   0683
                *string* ← "OPTION";                                    0684
                RETURN;                                                 0685
                END;                                                    0686
            ENDCASE;                                                    048
        RETURN (&resultptr );                                           049
        END.
                                                                        050
% READS AN REPEAT CHARACTER %                                           051
   (readrepeat) PROCEDURE( % parsing function which looks at the
   next input char.  If it is an REPEAT character, then it reads the
   char and returns TRUE, otherwise it returns FALSE %                  052
      % FORMAL ARGUMENTS %                                              053
        resultptr,          % ptr to the result record %                 054
        parsemode,          % parsing mode %                             055
        string);    % ptr to help string %                               056
      REF resultptr, string;                                            057
      % ------------- %                                                 058
        CASE parsemode OF                                               059
            = parsing:                                                  060
                CASE lookc() OF                                         061
                    = rptchar:                                          062
                        inpt();                                         063
                    ENDCASE RETURN (FALSE);                             064
            = parsehelp:                                                065
                *string* ← "RPT:";                                      066
            = parseqmark:                                               0687
                BEGIN                                                   0688
                *string* ← "OKREPEAT";                                  0689
                RETURN;                                                 0690
                END;                                                    0691
            ENDCASE;                                                    067
        RETURN (&resultptr );                                           068
        END.
                                                                        069
% READS AN ANSWER CHARAC ER %                                           070
   (answ) PROCEDURE( % reads next input char, returns TRUE if answer
   is yes, FALSE otherwise.  CA and Y denote YES, all other chars
   denote NO %                                                          071
      % FORMAL ARGUMENTS %                                              072
        resultptr,          % ptr to result record %                     073
        parsemode,          % interpreter parsing mode %                 074
        stringptr);         % ptr to help string %                       075
      REF resultptr, stringptr, inpt;                                   076
      % ---------------- %                                              077
      CASE parsemode OF                                                 078
        = parsing:                                                      079
            BEGIN                                                       080
            cueflg ← FALSE;                                             081
```

```
            CASE lookc() OF                                                 082
               = cachar, = optchar, = rptchar, = inschar:                   083
                  BEGIN                                                     084
                  inpt();                                                   085
                  RETURN (&resultptr);                                      086
                  END;                                                      087
               = 'Y, ='y:                                                   088
                  BEGIN                                                     089
                  inpt();                                                   090
                  curchr ← CA;                                             0746
                  RETURN (&resultptr);                                      091
                  END;                                                      092
               = CD:                                                        093
                  SIGNAL(cmdelete);                                         094
               =BC, =BW:                                                    095
                  SIGNAL (popstate);                                        096
               ENDCASE                                                      097
                  BEGIN                                                    01084
                  needconfirm ← TRUE;                                      01083
                  inpt();                                                   098
                  END;                                                     01085
            RETURN (FALSE);                                                 099
            END;                                                           0100
         = parsehelp:                                                      0101
            IF resultptr.begnodeptr = resultptr.curnodeptr THEN            01077
               *stringptr* ← "Y/N:"                                        0102
            ELSE *stringptr* ← NULL;                                       01079
         = parseqmark:                                                     0692
            BEGIN                                                          0693
            *stringptr* ← "Y for yes", 0, "N for no";                      01081
            RETURN;                                                        0695
            END;                                                           0696
         ENDCASE;                                                          0103
      RETURN (&resultptr);                                                 0104
      END.                                                                 0105

   (answer) PROCEDURE( % reads next input char, returns pointer to
   string 0/1.  CA and Y denote YES, all other chars denote NO %           0106
      % FORMAL ARGUMENTS %                                                0107
         resultptr,          % ptr to result record %                     0108
         parsemode,          % interpreter parsing mode %                 0109
         stringptr);         % ptr to help string %                       0110
      REF resultptr, stringptr, inpt;                                      0111
      % ---------------- %                                                0112
      CASE parsemode OF                                                    0113
         = parsing:                                                        0114
            BEGIN                                                          0115
            cueflg ← FALSE;                                                0116
            needconfirm ← TRUE;                                            0985
            CASE inpt() OF                                                 0117
               = cachar, = optchar, = rptchar, = inschar:                  0118
                  BEGIN                                                    0119
                  resultptr ← TRUE;                                        0120
                  END;                                                     0121
               = 'Y, ='y:                                                 0747
                  BEGIN                                                    0748
```

```
                resultptr ← TRUE;                                    0749
                curchr ← CA;                                         0751
                END;                                                 0750
             = CD:                                                   0122
                SIGNAL(cmdelete);                                    0123
             =BC, =BW:                                               0124
                SIGNAL (popstate);                                   0125
             ENDCASE                                                 0126
                resultptr ← FALSE;                                   0127
          RETURN (&resultptr);                                       0128
          END;                                                       0129
       = parsehelp:                                                  0130
          *stringptr* ← "Y/N:";                                      0131
       = parseqmark:                                                 0697
          BEGIN                                                      0698
          *stringptr* ← "Y for yes", 0, "N for no";                  0699
          RETURN;                                                    0700
          END;                                                       0701
       ENDCASE;                                                      0132
    RETURN (&resultptr);                                             0133
    END.                                                             0134

% LOOKS FOR A STATUS CHARACTER %                                     0986
   (lookstat) PROCEDURE( % looks at next input character, if it is S
   then performs the substitute show status routine and swallows the
   s and returns true, Otherwise it does not swallow the character
   and returns false%                                                0987
      % FORMAL ARGUMENTS %                                          0988
         resultptr,          % ptr to result record %               0989
         parsemode,          % interpreter parsing mode %           0990
         stringptr);         % ptr to help string  in parsehelp and
         parseqmark modes, source entity in parsing mode%            0991
      REF resultptr, stringptr, inpt;                                0992
      % ---------------- %                                          0993
      CASE parsemode OF                                              0994
         = parsing:                                                  0995
            BEGIN                                                    0996
            cueflg ← FALSE;                                          0997
            CASE lookc() OF                                          0998
               = 'S, ='S:                                           01029
                  BEGIN                                             01030
                  inpt();                                           01031
                  substatus(&resultptr,1,&stringptr);               01035
                  RETURN (&resultptr);                              01036
                  END;                                              01034
               = CD:                                                01010
                  SIGNAL(cmdelete);                                 01011
               =BC, =BW:                                            01012
                  SIGNAL (popstate);                                01013
               ENDCASE                                              01014
                  NULL;                                             01015
            RETURN (FALSE);                                         01016
            END;                                                    01017
         = parsehelp:                                               01018
            *stringptr* ← "S:";                                     01019
         = parseqmark:                                              01020
```

```
            BEGIN                                                   01021
            *stringptr* ← "S for status";                           01022
            RETURN;                                                 01023
            END;                                                    01024
         ENDCASE;                                                   01025
      RETURN (&resultptr);                                          01026
      END.
                                                                    01027
% LOOKS FOR ANSWER CHARACTER %                                       0135
   (lookansw) PROCEDURE( % looks at next input char, returns TRUE if
   answer is yes, FALSE otherwise.  CA and Y denote YES, all other
   chars denote NO %                                                0136
      % FORMAL ARGUMENTS %                                          0137
         resultptr,          % ptr to result record %              0138
         parsemode,          % interpreter parsing mode %          0139
         stringptr);         % ptr to help string %                0140
      REF resultptr, stringptr, inpt;                               0141
      % ---------------- %                                         0142
      CASE parsemode OF                                             0143
         = parsing:                                                 0144
            BEGIN                                                   0145
            cueflg ← FALSE;                                         0146
            CASE lookc() OF                                         0147
               = cachar, = optchar, = rptchar, = inschar:           0148
                  BEGIN                                             0149
                  RETURN (&resultptr);                              0150
                  END;                                              0151
               = 'Y, ='y:                                           0152
                  BEGIN                                             0153
                  inpt();                                           0154
                  curchr ← CA;                                      0745
                  RETURN (&resultptr);                              0155
                  END;                                              0156
               = CD:                                                0157
                  SIGNAL(cmdelete);                                 0158
               =BC, =BW:                                            0159
                  SIGNAL (popstate);                                0160
               ENDCASE                                              0161
                  inpt();                                           0162
            RETURN (FALSE);                                         0163
            END;                                                    0164
         = parsehelp:                                               0165
            *stringptr* ← "Y/N:";                                   0166
         = parseqmark:                                              0702
            BEGIN                                                   0703
            *stringptr* ← "Y for yes", 0, "N for no";               0704
            RETURN;                                                 0705
            END;                                                    0706
         ENDCASE;                                                   0167
      RETURN (&resultptr);                                          0168
      END.
                                                                    0169
   (mylookansw) PROCEDURE( % locks at next input char, returns TRUE
   if answer is yes, FALSE otherwise.  CA and Y denote YES, all
   other chars denote NO %                                          01037
      % FORMAL ARGUMENTS %                                          01038
```

```
        resultptr,              % ptr to result record %                 01039
        parsemode,              % interpreter parsing mode %             01040
        stringptr);             % ptr to help string %                   01041
    REF resultptr, stringptr, inpt;                                       01042
    % ---------------- %                                                 01043
    CASE parsemode OF                                                     01044
      = parsing:                                                          01045
        BEGIN                                                             01046
        cueflg ← FALSE;                                                   01047
        CASE lookc() OF                                                   01048
          = cachar, = optchar, = rptchar, = inschar:                      01049
            BEGIN                                                         01050
            RETURN (&resultptr);                                          01051
            END;                                                          01052
          = 'Y, ='y:                                                      01053
            BEGIN                                                         01054
            inpt();                                                       01055
            curchr ← CA;                                                  01056
            RETURN (&resultptr);                                          01057
            END;                                                          01058
          = CD:                                                           01059
            SIGNAL(cmdelete);                                             01060
          =BC, =BW:                                                       01061
            SIGNAL (popstate);                                            01062
          ENDCASE                                                         01063
            NULL;                                                         01064
        RETURN (FALSE);                                                   01065
        END;                                                              01066
      = parsehelp:                                                        01067
        *stringptr* ← "Y/N:";                                             01068
      = parseqmark:                                                       01069
        BEGIN                                                             01070
        *stringptr* ← "Y for yes", 0, "N for no";                         01071
        RETURN;                                                           01072
        END;                                                              01073
    ENDCASE;                                                              01074
    RETURN (&resultptr);                                                  01075
    END.                                                                  01076

% READS A NON-CA CHARACTER %                                             0170
  (notca) PROC(curptr, parsemode, string);                                0171
    % notca looks at the next inpt character, if it is not a CA,
    then the character is read and a true return is taken.  If the
    next character is a CA, then it is not read, and FALSE is
    returned %                                                            0172
    REF curptr, string;                                                   0173
    %------------%                                                        0174
    CASE parsemode OF                                                     0175
      = parsing:                                                          0176
        CASE lookc() OF                                                   0177
          # cachar:                                                       0178
            inpt();                                                       0179
        ENDCASE RETURN (FALSE);                                           0180
      = parsehelp:                                                        0181
        *string* ← "#CA:";                                                0182
      = parseqmark:                                                       0707
```

```
            BEGIN                                                   0708
            *string* ← "NOT <CA>";                                  0709
            RETURN;                                                 0710
            END;                                                    0711
        ENDCASE;                                                    0183
        RETURN (&curptr);                                           0184
        END.
                                                                    0185
                                                                    0186
% READS A CA CHARACTER %                                           0187
    (readconfirm) PROC(curptr, parsemode, string);                  0188
        % readconfirm looks at the next inpt character, if it is a
        CA/REPEAT/INSERT, then it is read and a true return is taken
        else FALSE is returned %                                    0189
        REF curptr, string;                                         0190
        %------------%                                             0191
        CASE parsemode OF                                           0192
            = parsing:                                              0193
                CASE lookc() OF                                     0194
                    = cachar, = rptchar, = inschar:                 0195
                        BEGIN                                       0196
                        % stick ptr to castring into result record % 0197
                            curptr ← $castr;                        0198
                        % read over the CA %                        0199
                            inpt();                                 0200
                        END;                                        0201
                ENDCASE RETURN (FALSE);                             0202
            = parsehelp:                                            0203
                *string* ← "OK:";                                   0204
            = parseqmark:                                           0712
                BEGIN                                               0713
                *string* ← "OK";                                    0714
                RETURN;                                             0715
                END;                                                0716
        ENDCASE;                                                    0205
        RETURN (&curptr);                                           0206
        END.
                                                                    0207
                                                                    0208
    (readbug) PROC(curptr, parsemode, string);                      0209
        % readbug looks at the next inpt character, if it is a CA,
        then it is read and a true return is taken else FALSE is
        returned %                                                  0210
        REF curptr, string;                                         0211
        %------------%                                             0212
        CASE parsemode OF                                           0213
            = parsing:                                              0214
                CASE lookc() OF                                     0215
                    = cachar, = rptchar, = inschar:                 0216
                        BEGIN                                       0217
                        % stick ptr to castring into result record % 0218
                            curptr ← $castr;                        0219
                        % read over the CA %                        0220
                            inpt();                                 0221
                        END;                                        0222
                ENDCASE RETURN (FALSE);                             0223
```

```
        = parsehelp:                                                    0224
           IF nlmode = fulldisplay THEN *string* ← "B:"                 0225
           ELSE *string* ← "OK:";                                       0655
        = parseqmark:                                                   0717
           BEGIN                                                        0718
           IF nlmode = fulldisplay THEN *string* ← "BUG"                0719
           ELSE *string* ← "OK";                                        0722
           RETURN;                                                      0720
           END;                                                         0721
     ENDCASE;                                                           0226
     RETURN (&curptr);                                                  0227
     END.
                                                                        0228
                                                                        0229
% LOOK FOR A CA CHARACTER %                                             0230
  (lookconfirm) PROC(curptr, parsemode, string);                        0231
     % lookconfirm looks at the next inpt character, if it is a
     CA/REPEAT/INSERT, then a true return is taken else FALSE is
     returned %                                                         0232
     REF curptr, string;                                                0233
     %------------%                                                     0234
     CASE parsemode OF                                                  0235
        = parsing:                                                      0236
           CASE lookc() OF                                              0237
              = cachar, = rptchar, = inschar:                           0238
                 NULL;                                                  0239
           ENDCASE RETURN (FALSE);                                      0240
        = parsehelp:                                                    0241
           *string* ← "OK:";                                            0242
        = parseqmark:                                                   0723
           BEGIN                                                        0724
           *string* ← "OK";                                             0725
           RETURN;                                                      0726
           END;                                                         0727
     ENDCASE;                                                           0243
     RETURN (&curptr);                                                  0244
     END.
                                                                        0245
                                                                        0246
  (lookbug) PROC(curptr, parsemode, string);                            0247
     % lookbug looks at the next inpt character, if it is a CA,
     then a true return is taken else FALSE is returned %              0248
     REF curptr, string;                                                0249
     %------------%                                                     0250
     CASE parsemode OF                                                  0251
        = parsing:                                                      0252
           CASE lookc() OF                                              0253
              = cachar, = rptchar, = inschar:                           0254
                 NULL;                                                  0255
           ENDCASE RETURN (FALSE);                                      0256
        = parsehelp:                                                    0257
           IF nlmode = fulldisplay THEN *string* ← "B:"                 0258
           ELSE *string* ← NULL;                                        0654
        = parseqmark:                                                   0728
           BEGIN                                                        0729
           IF nlmode = fulldisplay THEN *string* ← "BUG"                0733
```

```
            ELSE *string* ← NULL;                                          0734
            RETURN;                                                        0731
            END;                                                           0732
         ENDCASE;                                                          0259
      RETURN (&curptr);                                                    0260
      END.
                                                                           0261
                                                                           0262
% LOOK FOR A NUMBER %                                                     0276
   (looknum) PROC(curptr, parsemode, string);                              0277
      % looknum looks at the next inpt character, if it is a digit,
      then a true return is taken else FALSE is returned %                 0278
      REF curptr, string;                                                  0279
      %------------%                                                       0280
      CASE parsemode OF                                                    0281
         = parsing:                                                        0282
            CASE lookc() OF                                                0283
               IN ['0, '9]:                                                0284
                  NULL;                                                    0285
            ENDCASE RETURN (FALSE);                                        0286
         = parsehelp:                                                      0287
            *string* ← "NUM:";                                             0288
         = parseqmark:                                                     0740
            BEGIN                                                          0741
            *string* ← "NUMBER";                                           0742
            RETURN;                                                        0743
            END;                                                           0744
      ENDCASE;                                                             0289
      RETURN (&curptr);                                                    0290
      END.
                                                                           0291
                                                                           0292
% dnls or tnls %                                                          0948
   (isdnls) PROC(curptr, parsemode, string);                               0949
      % return TRUE if DNLS %                                              0950
      REF curptr, string;                                                  0951
      %------------%                                                       0952
      CASE parsemode OF                                                    0953
         = parsing:                                                        0954
            IF nlmode = typewriter THEN RETURN (FALSE);                    0958
         ENDCASE;                                                          0966
      RETURN (&curptr);                                                    0967
      END.
                                                                           0968
                                                                           0969
   (istnls) PROC(curptr, parsemode, string);                               0970
      % return TRUE if TNLS %                                              0971
      REF curptr, string;                                                  0972
      %------------%                                                       0973
      CASE parsemode OF                                                    0974
         = parsing:                                                        0975
            IF nlmode = fulldisplay THEN RETURN (FALSE);                   0976
         ENDCASE;                                                          0977
      RETURN (&curptr);                                                    0978
      END.
                                                                           0979
```

```
                                                                        0980
% RETURNS TRUE OR FALSE %                                              0941
   (false) PROC(result, parsemode);                                     0942
      IF parsemode = parsing THEN RETURN(0)                             0943
      ELSE RETURN(result);                                              0944
      END.
                                                                        0945
   (true) PROC(result, parsemode);                                      0946
      RETURN(result); END.
                                                                        0947
%prompt for lsel, dsel, and ssel%                                       0812
   (prmptlsel) PROC(curptr, parsemode, string);                         0839
      % generate correct prompting and ? response for an LSEL
      (excluding the type of parameter to be selected). %               0840
      REF curptr, string;                                               0841
      %------------%                                                   0842
      CASE parsemode OF                                                 0843
         = parsehelp:                                                   0849
            IF curptr.begnodeptr = curptr.curnodeptr THEN               0904
               BEGIN                                                    0912
               IF nlmode = fulldisplay THEN *string* ← "B/T"            0850
               ELSE *string* ← "T";                                     0851
               IF inprompts = partprompts THEN                          0907
                  *string* ← *string*, ':                               0908
               ELSE                                                     0909
                  *string* ← *string*, "/[A]:";                         0910
               END;                                                     0911
         = parseqmark:                                                  0852
            BEGIN                                                       0853
            IF nlmode = fulldisplay THEN *string* ← "BUG", 0,
            "TYPEIN", 0, "OPTION ADDRESS", 0                            0854
            ELSE *string* ← "TYPEIN", 0, "OPTION ADDRESS", 0;           0855
            END;                                                        0857
         ENDCASE;                                                       0858
      RETURN (&curptr);                                                 0859
      END.
                                                                        0860
                                                                        0861
   (prmptdsel) PROC(curptr, parsemode, string);                         0879
      % generate correct prompting and ? response for an DSEL
      (excluding the type of parameter to be selected). %               0880
      REF curptr, string;                                               0881
      %------------%                                                   0882
      CASE parsemode OF                                                 0883
         = parsehelp:                                                   0884
            IF nlmode = fulldisplay THEN *string* ← "B/A:"              0885
            ELSE *string* ← "A:";                                       0886
         = parseqmark:                                                  0887
            BEGIN                                                       0888
            IF nlmode = fulldisplay THEN *string* ← "BUG", 0,
            "ADDRESS", 0                                                0889
            ELSE *string* ← "ADDRESS", 0;                               0890
            END;                                                        0891
         ENDCASE;                                                       0892
      RETURN (&curptr);                                                 0893
      END.
```

```
                                                                    0894
                                                                    0895
    (prmptssel) PROC(curptr, parsemode, string);                    0862
       % generate correct prompting and ? response for an SSEL
       (excluding the type of parameter to be selected). %          0863
       REF curptr, string;                                          0864
       %------------%                                              0865
       CASE parsemode OF                                            0866
          = parsehelp:                                              0867
             BEGIN                                                  0905
             IF nlmode = fulldisplay THEN *string* ← "B/A"          0868
             ELSE *string* ← "A";                                   0869
             IF inprompts = partprompts THEN                        0896
                *string* ← *string*, ':                             0899
             ELSE                                                   0898
                *string* ← *string*, "/[T]:";                       0897
             END;                                                   0906
          = parseqmark:                                             0870
             BEGIN                                                  0871
             IF nlmode = fulldisplay THEN *string* ← "BUG", 0,
             "ADDRESS", 0, "OPTION TYPEIN", 0                       0872
             ELSE *string* ← "ADDRESS", 0, "OPTION TYPEIN", 0;      0873
             END;                                                   0874
          ENDCASE;                                                  0875
       RETURN (&curptr);                                            0876
       END.                                                         0877
                                                                    0878
% BACKUP CONTROL FUNCTION %                                         0293
   (setbkup) PROCEDURE( % sets backup point for parse %             0294
      % FORMAL ARGUMENTS %                                          0295
         resultptr,     % ptr to result record %                   0296
         parsemode);     % parsing mode %                          0297
      REF resultptr;                                                0298
      %------------%                                               0299
      % this routine is a dummy and does nothing except mark the backup
      point for the command repeat function %                       0300
      RETURN( &resultptr );                                         0301
      END.                                                          0302
% KEYWORD SEQUENCE DATA STRUCTURE MANIPULATION ROUTINES %           0303
   (seqbinit) PROC( % initializes a keyword data structure %        0304
      % The sequence structure is allocated here and is initialized for
      subsequent store operations %                                 0305
      % FORMAL ARGUMENTS %                                          0306
         resultptr,     % ptr to the result record %                0307
         parsemode);    % parsing mode %                           0308
      REF resultptr;                                                0309
      %------------%                                               0310
      CASE parsemode OF                                             0311
         = parsing:                                                 0312
            BEGIN                                                   0313
            resultptr[1] ← getstring(255, $dspblk ); % allocate 51
            words %                                                 0314
            resultptr ← resultptr[1]+1; % set up address of block % 0315
            [resultptr] ← 0; % initialize count to zero %           0316
```

```
            END;                                                    0317
         = backup,                                                  0318
         = cleanup:                                                 0319
            % deallocate the sequence block %                       0320
               freestring( resultptr[1], $dspblk );                 0321
         ENDCASE;                                                   0322
      RETURN( &resultptr );                                         0323
      END.                                                          0324

   (seqbadd) PROC( % adds a keyword to a keyword data structure %   0325
      % FORMAL ARGUMENTS %                                          0326
         resultptr,     % ptr to the result record %                0327
         parsemode,     % parsing mode %                            0328
         seqbstrptr,    % ptr to sequence structure %               0329
         strptr);       % ptr to keyword string %                   0330
      LOCAL count, ptr;                                             0331
      REF resultptr, strptr, seqbstrptr, ptr;                       0332
      %------------%                                                0333
      CASE parsemode OF                                             0334
         = parsing:                                                 0335
            BEGIN                                                   0336
            &ptr ← seqbstrptr; % get ptr to structure %             0337
            count ← ptr; % current count %                          0338
            BUMP count;                                             0339
            ptr[count] ← strptr;                                    0340
            ptr ← count;                                            0341
            END;                                                    0342
         ENDCASE;                                                   0343
      RETURN( &resultptr );                                         0344
      END.                                                          0345

%abort command as though user typed CD%                             0765
   (abort)     %abort current command specification%
   PROCEDURE (resultptr, parsemode);                                0752
      %clear input buffer and simulate a CD%                        0753
      REF resultptr;                                                0754
      CASE parsemode OF                                             0755
         = parsing:                                                 0756
            BEGIN                                                   0757
            clrbuf(0);  %clear input buffer%                        0758
            curchr ← CD;                                            0759
            unput();                                                0760
            END;                                                    0761
         ENDCASE;                                                   0762
      RETURN( &resultptr );                                         0763
      END.                                                          0764

%cut back the pathstack%                                            0766
   (cutback)   %cut back the pathstack to the frame for setcutback%
   PROCEDURE (curptr, parsemode);                                   0767
      REF curptr;                                                   0769
      CASE parsemode OF                                             0770
         = parsing:                                                 0771
            SIGNAL(cutpathstk);                                     0801
         ENDCASE;                                                   0777
      RETURN( &curptr );                                            0778
```

```
    END.                                                                0779

    (setcutback)        %set cutbackstop  to point to current frame%
    PROCEDURE (curptr, parsemode);                                      0802
       REF curptr;                                                      0804
       CASE parsemode OF                                                0805
          = parsing:                                                    0806
             cutstop ← &curptr;                                         0807
          ENDCASE;                                                      0808
       RETURN( &curptr );                                               0809
       END.                                                             0810

% CLEAR NAME AREA %                                                    0346
   (clearname) PROC(curptr, parsemode, string);                         0347
       % clears the name area and returns TRUE %                       0348
       REF curptr, string;                                              0349
       %------------%                                                  0350
       CASE parsemode OF                                                0351
          = parsing:                                                    0352
             dn ($"");                                                  0353
          ENDCASE;                                                      0354
       RETURN (&curptr);                                                0355
       END.                                                             0356
                                                                        0357
% CLIST UTILITY %                                                      0358
   (clist)    PROCEDURE (type, fno1, fno2);  %build clist%             0359
       %This routine constructs the correspondence list according to the
       parameters passed as follows: See clhdr and clistr for format of
       clist.                                                           0360
          If either fno1 or fno2 are > 0 then only stid's               0361
          belonging to those files are used.                            0362
          if type = 0 then clhead.clcnt is set to zero.                 0363
          if type .A 1 = 1 then the csp's are used.                     0364
          if type .A 2 = 2 then the frozen list entries are             0365
          used.                                                         0366
          if type .A 4 = 4 then the return ring entries are used.       0367
          if type .A 8 = 8 then the markers are used.                   0368
          any combination of the above is valid.%                       0369
       %----------------%                                              0370
       LOCAL                                                            0371
          end1, entry1, cl, fz, hd, mk, last1, a, b, srr, frr, i, j, fn;
                                                                        0372
       LOCAL STRING fnam1[150], fnam2[150];                             0373
       REF entry1, cl, fz, mk, srr, frr, fn;                            0374
       clhead.clbuff ← &cl ← .clistb;                                   0375
       clhead.clcnt ← 0;                                                0376
       clhead.clfno1 ← fno1;                                            0377
       clhead.clfno2 ← fno2;                                            0378
       clhead.cltype ← type;                                            0379
       IF type = 0 THEN RETURN;                                         0380
       IF type .A 4 THEN %file return ring%                             0381
          BEGIN                                                         0382
          %do not trust file numbers--use file name%                    0383
          *fnam1* ← NULL;                                               0384
          *fnam2* ← NULL;                                               0385
```

```
    IF fno1 THEN filnam(fno1, $fnam1);                                  0386
    IF fno2 THEN filnam(fno2, $fnam2);                                  0387
    END;                                                                0388
  end1 ← (&entry1 ← $dpyarea) + dacnt*da1;                              0389
  UNTIL &entry1 >= end1 DO                                              0390
    BEGIN                                                               0391
    IF type .A 1 THEN %csp's%                                           0392
      BEGIN                                                             0393
      IF entry1.daexis AND NOT entry1.daempty THEN                      0394
        IF (fno1 = 0 AND fno2 = 0) OR
         (entry1.dacsp.stfile = fno1) OR
         (entry1.dacsp.stfile = fno2) THEN                              0395
          BEGIN                                                         0396
          cl.clst1 ← entry1.dacsp;                                      0397
          cl.clcc1 ← entry1.daccnt;                                     0398
          cl.clst2 ← endfil;                                            0399
          cl.clcc2 ← 1;                                                 0400
          &cl ← &cl + cll;                                              0401
          BUMP clhead.clcnt;                                            0402
          END                                                           0403
        ELSE NULL;                                                      0404
      END;                                                              0405
    IF type .A 2 THEN %frozen list%                                     0406
      BEGIN                                                             0407
      IF entry1.daexis AND NOT entry1.daempty AND &fz ←                 0408
      entry1.dafrz1 THEN                                                0409
        DO                                                              0410
          IF (fno1 = 0 AND fno2 = 0) OR
           (fz.fzstid.stfile = fno1) OR
           (fz.fzstid.stfile = fno2) THEN                               0411
            BEGIN                                                       0412
            cl.clst1 ← fz.fzstid;                                       0413
            cl.clcc1 ← cl.clcc2 ← 1;                                    0414
            cl.clst2 ← endfil;                                          0415
            &cl ← &cl + cll;                                            0416
            BUMP clhead.clcnt;                                          0417
            END                                                         0418
          ELSE NULL                                                     0419
        UNTIL (&fz ← fz.fznext) = 0;                                    0420
      END;                                                              0421
    IF type .A 4 THEN %file return ring%                                0422
      BEGIN                                                             0423
      IF entry1.daexis THEN                                             0424
        BEGIN                                                           0425
        IF &frr ← entry1.dalink THEN                                    0426
          FOR i ← frrlength(&frr) DOWN UNTIL <= 0 DO                    0427
            BEGIN                                                       0428
            &fn ← readfrring(&frr, i : &srr);                           0429
            IF (fno1 = 0 AND fno2 = 0)
            OR (fno1 AND a ← (*fn* = *fnam1*))
            OR (fno2 AND b ← (*fn* = *fnam2*)) THEN                     0430
              BEGIN                                                     0431
              FOR j ← srrlength(&srr) DOWN UNTIL <= 0 DO                0432
                BEGIN                                                   0433
                cl.clst1 ← readsrring(&srr, j : cl.clcc1);
                                                                        0434
```

```
                        cl.clst1.stfile ←                                   0435
                           IF a THEN fno1 ELSE fno2;                        0436
                        cl.clst2 ← endfil;                                  0437
                        cl.clcc2 ← 1;                                       0438
                        &cl ← &cl + cll;                                    0439
                        BUMP clhead.clcnt;                                  0440
                        END;                                                0441
                     END;                                                   0442
                  END                                                       0443
               ELSE err($"no file return ring in clist");                   0444
               END;                                                         0445
            END;                                                            0446
         IF type .A 8 THEN %markers%                                        0447
            BEGIN                                                           0448
            IF entry1.daexis AND NOT entry1.daempty THEN                    0449
               IF  (fno1 = 0 AND fno2 = 0) OR
               (entry1.dacsp.stfile = fno1) OR                              0450
               (entry1.dacsp.stfile = fno2) THEN                            0451
                  BEGIN                                                     0452
                  &mk ← $mkrtb - $filhed + (hd ←                            0453
                     filhdr(entry1.dacsp.stfile));                          0454
                  last1 ← &mk + [$mkrtbl - $filhed + hd] * mkrl;            0455
                  %bounds check%                                            0456
                     IF (last1 - &mk) > [$mkrtxn + hd - $filhed] THEN
                                                                            0457
                        BEGIN                                               0458
                        [$mkrtbl - $filhed + hd] ← [$mkrtxn + hd -
                        $filhed] / mkrl;                                    0459
                        last1 ← &mk + [$mkrtbl - $filhed + hd] * mkrl;
                                                                            0460
                        END;                                                0461
                  UNTIL &mk >= last1 DO                                     0462
                     BEGIN                                                  0463
                     cl.clst1 ← 0;                                          0464
                     cl.clst1.stfile ← entry1.dacsp.stfile;                 0465
                     cl.clst1.stpsid ← mk.mkpsid;                           0466
                     cl.clcc1 ← mk.mkccnt;                                  0467
                     cl.clst2 ← endfil;                                     0468
                     cl.clcc2 ← 1;                                          0469
                     &cl ← &cl + cll;                                       0470
                     BUMP clhead.clcnt;                                     0471
                     &mk ← &mk + mkrl;                                      0472
                     END;                                                   0473
                  END                                                       0474
               ELSE NULL;                                                   0475
            END;                                                            0476
         &entry1 ← &entry1 + dal;                                           0477
         END;                                                               0478
      IF clhead.clcnt > clmax THEN err($"NLS system error");                0479
      RETURN;                                                               0480
      END.
                                                                            0481
(clupdt)   PROCEDURE;  %unbuild clist%                                      0482
      %This routine updates various things from the correspondence list
      according to the parameters in the clist header (clhead):             0483
         If either clhead.clfno1 or clhead.clfno2 are > 0 then              0484
```

```
    only stid's belonging to those files are updated.                    0485
    if cltype = 0 then none of the following is changed.                 0486
    if cltype .A 1 = 1 then the csp's are updated.                       0487
    if cltype .A 2 = 2 then the frozen lists are updated.                0488
    if cltype .A 4 = 4 then the return ring is updated.                  0489
    if cltype .A 8 = 8 then the markers are updated.                     0490
    any combination of the above is valid.                               0491
see clhdr and clistr for format of clist.%                               0492
    % if clst1  endfil, then this statement has been deleted from
    the file and clst2 points to the "next" statement in the file.
    if stccl changes then the text that used to be pointed to has
    been deleted or moved and clccl has been updated to something
    reasonable.%                                                         0914
%----------------%                                                       0493
LOCAL                                                                    0494
    endl, stid, entryl, cl, fz, fz2, hd, viewl, view2,                   0495
    mklngth, mkold, mk, lastl, srr, frr, fn, i, j, cc, a, b;             0496
LOCAL TEXT POINTER bl;                                                   0913
LOCAL STRING fnaml[150], fnam2[150];                                     0497
REF entryl, cl, fz, fz2, mklngth, mk, srr, frr, fn;                      0498
IF clhead.cltype = 0 THEN RETURN;                                        0499
&cl ← clhead.clbuff;                                                     0500
IF clhead.cltype .A 4 THEN %link1 ring%                                  0501
    BEGIN                                                                0502
    %do not trust file numbers--use file name%                           0503
    *fnaml* ← NULL;                                                      0504
    *fnam2* ← NULL;                                                      0505
    IF clhead.clfnol THEN filnam(clhead.clfnol, $fnaml);                 0506
    IF clhead.clfno2 THEN filnam(clhead.clfno2, $fnam2);                 0507
    END;                                                                 0508
endl ← (&entryl ← @dpyarea) + dacnt*dal;                                 0509
UNTIL &entryl = endl DO                                                  0510
    BEGIN                                                                0511
    IF clhead.cltype .A 1 THEN %csp's%                                   0512
        BEGIN                                                            0513
        IF entryl.daexis AND NOT entryl.daempty THEN                     0514
            IF (clhead.clfnol= 0 AND clhead.clfno2= 0) OR
            (clhead.clfnol AND entryl.dacsp.stfile = clhead.clfnol)
            OR
            (clhead.clfno2 AND entryl.dacsp.stfile = clhead.clfno2)
            THEN                                                         0515
                BEGIN                                                    0516
                IF cl.clstl = endfil OR cl.clstl.stfile NOT=             0517
                entryl.dacsp.stfile THEN %replace by clst2%              0518
                    BEGIN                                                0519
                    IF cl.clst2 = endfil THEN                            0915
                        BEGIN                                            0916
                        entryl.dacsp ← endfil;                           0917
                        entryl.daccnt ← 1;                               0918
                        entryl.daempty ← TRUE;                           0919
                        END                                              0920
                    ELSE                                                 0921
                        BEGIN                                            0922
                        bl ← cl.clst2;                                   0520
                        bl[1] ← cl.clcc2;                                0521
                        IF POS bl >= SE(bl) THEN FIND SE(bl) CH ↑bl;
```

```
                                                                    0522
                    entry1.dacsp ← b1;                              0523
                    entry1.daccnt ← b1[1];                          0524
                    END;                                            0923
                 END                                                0525
              ELSE                                                  0526
                 BEGIN                                              0527
                 IF entry1.dacsp NOT= cl.clst1 THEN                 0528
                    entry1.dacsp ← cl.clst1;                        0924
                 IF entry1.daccnt NOT= cl.clcc1 THEN                0529
                    entry1.daccnt ← cl.clcc1;                       0925
                 END;                                               0530
              &cl ← &cl + cll;                                      0531
              END;                                                  0532
        END;                                                        0533
     IF clhead.cltype .A 2 THEN %frozen list%                       0534
        BEGIN                                                       0535
        IF entry1.daexis AND NOT entry1.daempty AND &fz ←           0536
        entry1.dafrz1 THEN                                          0537
           DO                                                       0538
              IF (clhead.clfno1= 0 AND clhead.clfno2= 0) OR
              (clhead.clfno1 AND fz.fzstid.stfile = clhead.clfno1)
              OR
              (clhead.clfno2 AND fz.fzstid.stfile = clhead.clfno2)
              THEN                                                  0539
                 BEGIN                                              0540
                 IF cl.clst1 = endfil THEN                          0541
                    BEGIN %statement was deleted%                   0542
                    IF (&fz2 ← entry1.dafrz1) = &fz THEN            0543
                       entry1.dafrz1 ← fz.fznext                    0544
                    ELSE                                            0545
                       DO IF fz2.fznext = &fz THEN                  0546
                          BEGIN                                     0547
                          fz2.fznext ← fz.fznext;                   0548
                          EXIT;                                     0549
                          END                                       0550
                       UNTIL (&fz2 ← fz2.fznext) = 0;               0551
                    &fz2 ← fz.fznext;                               0926
                    fz.fznext ← fzfree := &fz;                      0552
                    &fz ← &fz2;                                     0927
                    IF &fz = 0 THEN EXIT LOOP ELSE REPEAT LOOP;     0928
                    END                                             0553
                 ELSE                                               0554
                    IF cl.clst1 # fz.fzstid THEN                    0555
                       fz.fzstid ← cl.clst1;                        0556
                 &cl ← &cl + cll;                                   0557
                 END                                                0558
           UNTIL (&fz ← fz.fznext) = 0;                             0559
        END;                                                        0560
     IF clhead.cltype .A 4 THEN %file return ring%                  0561
        BEGIN                                                       0562
        IF entry1.daexis THEN                                       0563
           BEGIN %cycle through frr and srr's%                      0564
           IF &frr ← entry1.dalink THEN                             0565
              FOR i ← frrlength(&frr) DOWN UNTIL <= 0 DO            0566
                 BEGIN                                              0567
```

```
                &fn ← readfrring(&frr, i : &srr);                    0568
                IF (clhead.clfno1 = 0 AND clhead.clfno2 = 0)
                OR (clhead.clfno1 AND a ← (*fn* = *fnam1*))
                OR (clhead.clfno2 AND b ← (*fn* = *fnam2*)) THEN
                                                                     0569
                   BEGIN %found a match -- update stid, cc for
                   each entry in srr%                                0570
                   FOR j ← srrlength(&srr) DOWN UNTIL <= 0 DO        0571
                      BEGIN                                          0572
                      stid ← readsrring(&srr, j : cc, view1,
                      view2);                                        0573
                      IF cl.clst1 NOT= endfil THEN                   0574
                         BEGIN                                       0575
                         IF stid.stpsid NOT= cl.clst1.stpsid OR cc
                         NOT= cl.clcc1 THEN                          0576
                            storesrring(&srr, j, cl.clst1,
                            cl.clcc1, view1, view2)                  0577
                         END                                         0578
                      ELSE                                           0579
                         storesrring(&srr, j, cl.clst2, cl.clcc2,
                         view1, view2);                              0580
                      &cl ← &cl + cll;                               0581
                      END;                                           0582
                   END;                                              0583
                END                                                  0584
             ELSE err($"no file return ring in clupdt");             0585
             END;                                                    0586
          END;                                                       0587
       IF clhead.cltype .A 8 THEN %markers%                          0588
          BEGIN                                                      0589
          IF entry1.daexis AND NOT entry1.daempty THEN               0590
             IF (clhead.clfno1= 0 AND clhead.clfno2= 0) OR
             (clhead.clfno1 AND entry1.dacsp.stfile= clhead.clfno1)
             OR
             (clhead.clfno2 AND entry1.dacsp.stfile= clhead.clfno2)
             THEN                                                    0591
                BEGIN                                                0592
                &mk ← mkold ← $mkrtb - $filhed + (hd ←               0593
                   filhdr(clhead.clfno1));                           0594
                last1 ← &mk + (&mklngth ← $mkrtbl - $filhed          0595
                + hd)/*mkr1;                                         0596
                UNTIL &mk = last1 DO                                 0597
                   BEGIN                                             0598
                   IF cl.clst1 = endfil THEN %delete marker%         0607
                      BEGIN                                          0608
                      mvbfbf(&mk + mkr1, &mk,                        0609
                      last1-&mk-mkr1);                               0610
                      BUMP DOWN mklngth;                             0611
                      last1 ← last1 - mkr1;                          0612
                      &cl ← &cl + cll;                               0933
                      END                                            0613
                   ELSE                                              0614
                      BEGIN                                          0929
                      IF mk.mkpsid NOT= cl.clst1.stpsid THEN         0931
                         mk.mkpsid ← cl.clst1.stpsid;                0599
                      IF mk.mkccnt NOT= cl.clcc1 THEN                0932
```

```
                        mk.mkccnt ← cl.clccl;                          0601
                     &cl ← &cl + cll;                                  0602
                     &mk ← &mk + mkrl;                                 0603
                     END;                                              0930
                  END;                                                 0604
               END;                                                    0615
            END;                                                       0616
         &entryl ← &entryl + dal;                                      0617
         END;                                                          0618
      RETURN;                                                          0619
      END.
                                                                       0620
   (dpset) PROCEDURE (option, stidl, stid2, stopstid);                 0621
      %set global variables for recreate display routines.  In some
      cases, stidl and stid2 are passed merely to indicate file
      involvement, not because they will be reformatted.%              0622
      cdtype ← option;                                                 0623
      IF stidl = stid2 THEN stid2 ← endfil;                            0657
      (cdstdl, cdstd2) ← (stidl, stid2);                               0624
      IF option = dsprmt THEN %save statements before edit%            0656
         BEGIN                                                         0658
         IF stidl NOT= endfil THEN                                     0662
            *cdstrl* ← $SF(stidl) SE(stidl);                           0659
         IF stid2 NOT= endfil THEN                                     0663
            *cdstr2* ← $SF(stid2) SE(stid2);                           0661
         END;                                                          0660
      IF stopstid.stpsid = orgstid THEN cdstop ← endfil                0625
      ELSE cdstop ← stopstid;                                          0626
      RETURN END.                                                      0627
                                                                       0628
   (dpstp) PROCEDURE (stid);                                           0629
      %called by text editing control routines to accept the stid of a
      bugged statement as inpt and return an appropriate stid for the
      display parameter CDSTOP (an stid on the same level or higher
      which appears after 'stid' in the file).%                        0630
      UNTIL NOT <FILMNP, getitl> (stid)                                0631
         DO stid ← <FILMNP, getsuc> (stid);                            0632
      RETURN (stid ← getsuc (stid));                                   0633
      END.                                                             0634
                                                                       0635
% NUMBER CONVERSION %                                                 0636
   (getpint)  % convert 2 text pointers to integer %                   0637
      PROCEDURE                                                        0638
         (tp1,          % starting text pointer %                     0639
         tp2 );         % ending text pointer %                        0640
      LOCAL STRING                                                     0641
         locstr[50];    % temp string %                                0642
      REF tp1, tp2;                                                    0643
                                                                       0644
      *locstr* ← tp1 tp2;                                              0645
      FIND SF(*locstr*);                                               0646
      CASE READC OF                                                    0647
         = D ; REPEAT CASE;                                            0648
         = ENDCHR ; EXIT CASE;                                         0649
         ENDCASE err( $"Illegal Number" );                             0650
      RETURN(VALUE($locstr));                                          0651
```

```
    END.                                                                0652
% give warning message if experimental system %                         0664
   (xwarning) PROC(curptr, parsemode);                                  0665
      % display a warning message to the user if he is using the
      experimental system and returns TRUE %                            0666
      REF curptr;                                                       0667
      %------------%                                                    0668
      CASE parsemode OF                                                 0669
         = parsing:                                                     0670
            IF jdebug THEN                                              0671
               dismes (2, $"WARNING: EXPERIMENTAL SYSTEM, use at your
               own risk!");                                             0676
         ENDCASE;                                                       0672
      RETURN (&curptr);                                                 0673
      END.
                                                                        0674
                                                                        0675
FINISH                                                                  0653
```

KUNLUK

(MLK)RUNLDR (MLK)RUNLDR (MLK)RUNLDR (MLK)RUNLDR (MLK)RUNLDR (M
(MLK)RUNLDR (MLK)RUNLDR (MLK)RUNLDR (MLK)RUNLDR (MLK)RUNLDR (M

```
< NLS, RUNLDR.NLS;32, >, 3-OCT-74 17:08 KEV ;;;;
;Output Assembler File (rel-nls,runldr.txt,)                    03
CONN REL-NLS                                                   02
DA<ESC>                                                        04
TENLDR                                                         05
/S                                                             06
/377777S                                                       07
/M                                                             08
/117E                                                          09
/4000100                                                       010
PDATA                                                          011
SYNTAX                                                         012
SYNTBL                                                         095
PARSER                                                         013
SELECT                                                         014
PSUPPORT                                                       015
PSEDIT                                                         016
PSCALC                                                         098
PSSENDMAIL                                                     017
PSPROGS                                                        018
PSSYSTEM                                                       088
PSUSEROP                                                       024
PSHELP                                                         020
PSSYNGEN                                                       096
FONLY                                                          077
RESOLVER                                                       025
/c resolver                                                    026
BINTNLS                                                        084
SINTNLS                                                        087
FINTNLS                                                        062
UPOVER                                                         073
/5520000                                                       074
UPDATA                                                         075
/1400                                                          031
BDATA                                                          078
FDATA                                                          032
SDATA                                                          086
L10DATA                                                        079
NDDTDATA                                                       080
UNDATA                                                         085
FCONST                                                         081
BCONST                                                         033
FRECORDS                                                       034
BRECORDS                                                       082
SRECORDS                                                       083
L10RUNTIME                                                     089
UTILTY                                                         035
CEDIT                                                          036
FILMNP                                                         037
SEQGEN                                                         038
DSPGEN                                                         039
TSPRT                                                          040
EXECFL                                                         041
FRONTEND                                                       030
PRMSPC                                                         028
ADRMNP                                                         029
```

```
IOEXEC                                                          042
STGMGT                                                          043
CORESUPPORT                                                     044
VERIFY                                                          045
SEQFIL                                                          046
AUXCOD                                                          047
OPTAB                                                           049
NDDT                                                            050
NDDT2                                                           051
COLSRT                                                          048
RECFIL                                                          055
IDENTSUPPORT                                                    057
CSENDMAIL                                                       058
CPROGRAMS                                                       059
CHELP                                                           061
CALCSUPPORT                                                     094
CSYNGEN                                                         097
INPFBK                                                          027
CATNUM                                                          052
<SYSTEM>STENEX                                                  063
/120R                                                           064
<ESC>                                                           065
DDT                                                             091
DDT+1<ESC>3B QT<ESC>GNNNXNLS.SAV;<ESC>                          092
                                                                068
QUIT                                                            071
```

SDATA

(MLK)SDATA (MLK)SDATA (MLK)SDATA (MLK)SDATA (MLK)SDATA (MLK)SD
(MLK)SDATA (MLK)SDATA (MLK)SDATA (MLK)SDATA (MLK)SDATA (MLK)SD

```
< NLS. SDATA.NLS;6, >, 11-OCT-74 10:43 KEV ;;;;
FILE sdata   %  LIO   <REL-NLS>SDATA   %% (LIO,) (rel-nls,SDATA.rel,) %
                                                                      02
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
%Identification system%                                               06
   DECLARE EXTERNAL STRING                                            07
      userstr[50]; %string corresponding to login dir name% %PASSED%  08
% DECLARE's %                                                         09
   DECLARE EXTERNAL                                                  010
      entvec[7], %entry vector for nls%                              011
      tenex,  %tenex version number%                                 012
      lhostn,  %logical host number on network%                      013
   %...illegal instruction pseudo-interrupt support...%              014
      ilsdsp,  %dispatch address%                                    015
      %DO NOT REORDER THE NEXT TWO DECLARATIONS%                     016
      psireg[15], psirglast;  %registers at time of psi%             017
   DECLARE EXTERNAL TEXT POINTER                                     018
      p1, p2;                                                        019
   DECLARE EXTERNAL                                                  020
   %...storage management...%                                        026
       cgetblk, cfreeblk, cmakezone, creplenish, cunlink, clink,     027
   %...pseudointerrupt tables...%       %PASSED%                     028
      levtab=lev1i#, levt2=lev2ic, levt3=lev3ic,                     029
      lev1ic,                                                        030
      lev2ic,                                                        031
      lev3ic,                                                        032
      chntab[36],                                                    033
      savchntab[4], %for use in no-oping first four pseudo interrupts%
                                                                     034
      trpcnt = 0,                                                    035
      ccignore,  %count of how many times to ignore control characters%
                                                                     036
   %...pseudointerrupt data...%         %PASSED%                    037
      sav40,                                                         052
      svac1, svacs[15], svacse,                                      038
   %...system timer globals...%                                      039
      timrset=0,   %timer is running flag%                           040
      timrproc,   %procedure to be called when the timmer goes off%  041
      timra1,   % argument 1 for timrproc %                          042
      timra2,   % argument 2 for timrproc %                          043
      timra3,   % argument 3 for timrproc %                          044
      timra4,   % argument 4 for timrproc %                          045
%...global display/print parameters...%                              046
   %...display support...%                                           047
      msglck,                                                        048
      msgtim,                                                        049
      msgfrk;                                                        050
% for use with syntax generating commands %                          053
   % stack and stack pointers for partial path generation %          065
      DECLARE EXTERNAL csstkx, csstkb, csstk[100];                   066
   % variables controlling output mode %                            054
      DECLARE EXTERNAL                                               055
         omode = 0,    % output modes %                              056
            % 0 - debugging: do typeas in onode %                    057
```

```
        % 1 - inserting: do insert statement in onode %                  058
          % cinstid = stid of statement to insert after %                059
          % ilevel = level to insert at %                               060
        % 2 - being used as seqgen: do send in onode %                   061
          % instid = adr of seq work area %                             062
        % 3 - being used in show type command: call fbctl in onode
        %                                                                063
      cinstid, ilevel;                                                   064

FINISH of sdata                                                          051
```

SELECT

(MLK)SELECT (MLK)SELECT (MLK)SELECT (MLK)SELECT (MLK)SELECT (M

```
< NLS, SELECT.NLS;60, >, 23-OCT-74 12:53 DSM ;;;;
FILE select % L10 <rel-nls>select %% (l10,) (rel-nls,select.rel,) %    02
   % DECLARATIONS %                                                    03
      REF tda, inpt, fbstr, sysmsg;                                    04
   % SELECTION PROCESSORS %                                           047
      % SELECTION ROUTINES %                                          048
         (xselect) PROCEDURE(                                        049
            % process a selection %                                   050
            % FORMAL ARGUMENTS %                                      051
               resultptr,       % ptr to the return argument record % 052
               parsemode,       % parsing mode %                      053
               type,    % address of type code for selection %       054
               curptr );        % ptr to path stack entry %           055
            % NORMAL RETURNS %                                        056
               % 2 text pointers are returned in the result record
               which delemit the selection %                         057
            % ABNORMAL RETURNS %                                      058
               % a signal is generated whenever a CD is encounterd % 059
            LOCAL % VARIABLES %                                      060
               bugaction,               % what to do when a BUG is typed
               %                                                     061
               bugs,            % number of bug selections required % 062
               caflag,                  % TRUE if CA requried after help
               %                                                     063
               char,            % inpt char %                       064
               defaction,               % what to do on typing any other
               char %                                                065
               delimiter,               % entity delimiter routine %  066
               function,                % processing function %       067
               i,                       % number of selections obtained
               %                                                     068
               ltda,            % ptr to current display area %     069
               oaction,                 % what to do when a OPT char is
               typed %                                               070
               optionstr,               % str containing current options
               %                                                     071
               promptsav,               % ptr to prompts saving string %
                                                                     072
               statesav,                % ptr to state saving area %  073
               temp,            % scratch variable %                074
               tptr;            % ptr to a result textptr %         075
            LOCAL STRING                                             076
               helpstr[200];    % help feedback string %            077
            REF % VARIABLES %                                        078
               bugaction,                                            079
               curptr,                                               080
               defaction,                                            081
               delimiter,                                            082
               function,                                             083
               ltda,                                                 084
               oaction,                                              085
               promptsav,                                            086
               resultptr,                                            087
               statesav,                                             088
               tptr,                                                 089
               type;                                                 090
```

```
REF tda, inpt, fbstr, sysmsg;                                   02573
%-----------------%                                               091
% trap error signals in order to recover from selection
errors without aborting the command %                             092
   ON SIGNAL                                                      093
      = errsig:                                                   094
        BEGIN                                                     095
        % output the error message %                              096
           dismes( 2, MESSAGE);                                   097
           &sysmsg ← $"";                                         098
        % invoke the xselect routine in cleanup mode to
        reset any bugs, etc that it may have put up %             099
           xselect( &resultptr, cleanup);                         0100
        % reset the feedback %                                    0101
           cflstr.L ← statesav.cfllen;                            0102
           fbctl( fbpop );                                        0103
        % reset selection count %                                 0104
           statesav.nselects ← 0;                                 0105
        % resume gathering of a selection %                       0106
           GOTO selectagain;                                      0107
        END;                                                      0108
   ELSE;                                                          0109
(selectagain):                                                    0110
% set ptr to selection state record %                            0111
   &statesav ← &resultptr + pselLen;                              0112
% clear the needconfirm flag %                                   0113
   needconfirm ← FALSE;                                           0114
% clear special control character $echo flag %                   01645
   ctrlchar ← FALSE;                                              01646
% process according to parsemode %                               0115
   CASE parsemode OF                                              0116
      = parsing:   % normal parsing mode %                        0117
        BEGIN                                                     0118
        slink ← cdlnk ← clpsw ← nwlnk ← FALSE;                    02227
        CASE type OF                                              0119
           % STRUCTURAL ENTITIES %                                0120
              = $branch:                                          0121
                 BEGIN                                            0122
                 bugs ← 1;                                        0123
                 &delimiter ← $stdr;                              0124
                 END;                                             0125
              = $group:                                           0126
                 BEGIN                                            0127
                 bugs ← 2;                                        0128
                 &delimiter ← $grpdr;                             0129
                 END;                                             0130
              = $plex:                                            0131
                 BEGIN                                            0132
                 bugs ← 1;                                        0133
                 &delimiter ← $plxdr;                             0134
                 END;                                             0135
              = $statement:                                       0136
                 BEGIN                                            0137
                 bugs ← 1;                                        0138
                 &delimiter ← $stdr;                              0139
                 END;                                             0140
```

```
                % TEXTUAL ENTITIES %                                    0141
                  = $character:                                        0142
                     BEGIN                                             0143
                     &delimiter ← $cdr;                                0144
                     bugs ← 1;                                         0145
                     END;                                              0146
                  = $controlchar:                                     01638
                     BEGIN                                            01639
                     &delimiter ← $tdr;                               01640
                     bugs ← 2;                                        01641
                     ctrlchar ← TRUE; % special echos %               01642
                     END;                                             01643
                  = $invisible:                                        0147
                     BEGIN                                             0148
                     &delimiter ← $idr;                                0149
                     bugs ← 1;                                         0150
                     END;                                              0151
                  = $link:                                             0152
                     BEGIN                                             0153
                     slink ← TRUE;                                    02222
                     cdlnk ← TRUE;                                    02225
                     &delimiter ← $ldr;                                0154
                     bugs ← 1;                                         0155
                     END;                                              0156
                  = $directory:                                       02229
                     BEGIN                                            02230
                     slink ← TRUE;                                    02234
                     &delimiter ← $vdr;                               02231
                     bugs ← 1;                                        02232
                     END;                                             02233
                  = $password:                                        02235
                     BEGIN                                            02236
                     clpsw ← TRUE;                                    02241
                     &delimiter ← $vdr;                               02238
                     bugs ← 1;                                        02239
                     END;                                             02240
                  = $number:                                           0157
                     BEGIN                                             0158
                     &delimiter ← $ndr;                                0159
                     bugs ← 1;                                         0160
                     END;                                              0161
                  = $text:                                             0162
                     BEGIN                                             0163
                     &delimiter ← $tdr;                                0164
                     bugs ← 2;                                         0165
                     END;                                              0166
                  = $visible:                                          0167
                     BEGIN                                             0168
                     &delimiter ← $vdr;                                0169
                     bugs ← 1;                                         0170
                     END;                                              0171
                  = $word:                                             0172
                     BEGIN                                             0173
                     &delimiter ← $wdr;                                0174
                     bugs ← 1;                                         0175
                     END;                                              0176
```

```
                % MISC. ENTITIES %                                    0177
                    = $file:                                         0178
                       BEGIN                                         0179
                       slink ← cdlnk ← TRUE;                         03371
                       &delimiter ← $flndr;                          0180
                       bugs ← 1;                                     0181
                       END;                                          0182
                    = $newfilelink:                                  02249
                       BEGIN                                         02250
                       slink ← cdlnk ← nwlnk ← TRUE;                 02257
                       &delimiter ← $ldr;                            02253
                       bugs ← 1;                                     02254
                       END;                                          02255
                    = $oldfilelink:                                  02242
                       BEGIN                                         02243
                       slink ← cdlnk ← TRUE;                         02259
                       &delimiter ← $ldr;                            02246
                       bugs ← 1;                                     02247
                       END;                                          02248
                    = $name:                                         0188
                       BEGIN                                         0189
                       &delimiter ← $nmdr;                           0190
                       bugs ← 1;                                     0191
                       END;                                          0192
                    = $ident:                                        01694
                       BEGIN                                         01695
                       &delimiter ← $iddr;                           01696
                       bugs ← 1;                                     01697
                       END;                                          01698
                    = $identlist:                                    01699
                       BEGIN                                         01700
                       &delimiter ← $idldr;                          01701
                       bugs ← 2;                                     01702
                       END;                                          01703
                    = $edge:                                         0193
                       BEGIN                                         0194
                       &delimiter ← 0;                               0195
                       bugs ← 1;                                     0196
                       END;                                          0197
                    = $marker:                                       0198
                       BEGIN                                         0199
                       &delimiter ← $wdr;                            0200
                       bugs ← 1;                                     0201
                       END;                                          0202
                ENDCASE SIGNAL (interperr, $"Unknown Entity
                selection type");                                    0203
                % we now set up actions to be accompilshed
                whenever a trigger character is encountered .  The
                characters currently recognized as trigger
                characters are :                                     0204
                   1) CA character-- may be a bug selection.         0205
                   2) $option char-- indicated what to do when an
                   $option is typed.                                 0206
                   3) '? character:  request for some prompting.
                                                                     0207
                   4) any other char: --this is the default action
```

```
for the type of selection. %                                    0208
% the action function is set to 0 if it is not
permitted %                                                     0209
caflag ← FALSE;                                                 0210
CASE /curptr.curnodeptr/.opcode OF                              0211
   = $dsel:                                                     0212
      BEGIN                                                    02636
      &defaction ← $getdae;                                     0225
      &oaction ← 0;                                             0227
      IF nlmode = fulldisplay THEN                              0213
         BEGIN                                                  0215
         IF type = $edge THEN                                  01538
            BEGIN                                              01541
            &defaction ← 0;                                     0216
            &bugaction ← $getwindow;                           01540
            optionstr ← $"BUG";                                01544
            END                                                01542
         ELSE                                                  01545
            BEGIN                                              01546
            &bugaction ← $getbug;                               0217
            optionstr ←  $"BUG   or   ADDRESS";
                                                                0219
            END;                                               01547
         caflag ← TRUE;                                         0220
         arm();                                                 0221
         END                                                    0222
      ELSE                                                      0223
         BEGIN                                                  0224
         &bugaction ← $getdae;                                  0226
         optionstr ← $"ADDRESS";                                0228
         END;                                                   0229
      END;                                                     02638
   = $ssel:                                                     0230
      BEGIN                                                    02637
      &defaction ← $getdae;                                     0234
      &oaction ← (CASE type OF                                 02755
         = $ident: $getid;                                     02756
         = $identlist: $getidlist;                             02757
         ENDCASE $getlit);                                     02758
      IF nlmode = fulldisplay THEN                              0231
         BEGIN                                                  0233
         &bugaction ← $getbug;                                  0235
         optionstr ← $"BUG   or   ADDRESS   or
          OPTION TYPEIN";                                       0237
         caflag ← TRUE;                                         0238
         arm();                                                 0239
         END                                                    0240
      ELSE                                                      0241
         BEGIN                                                  0242
         &bugaction ← $getdae;                                  0244
         optionstr ← $"ADDRESS  or  OPTION
         TYPEIN";                                               0246
         END;                                                   0247
      END;                                                     02639
   = $lsel:                                                     0248
      BEGIN                                                     0249
```

```
                &defaction ← (CASE type OF                          0252
                   = &ident: $getid;                                02752
                   = &identlist: $getidlist;                        02753
                   ENDCASE $getlit);                                02754
                &oaction ← $getdae;                                  0254
                IF nlmode = fulldisplay THEN                         0250
                   BEGIN                                             02632
                   &bugaction ← $getbug;                             0251
                   optionstr ← $"TYPEIN   or   BUG   or
                   OPTION ADDRESS";                                  02628
                   caflag ← TRUE;                                    02629
                   arm();                                            02630
                   END                                               02631
                ELSE                                                 02627
                   BEGIN                                             02633
                   &bugaction ← $getlit;                             0253
                   optionstr ← $"TYPEIN   or   OPTION
                   ADDRESS";                                         0255
                   END;                                              02634
                END;                                                 0256
             ENDCASE SIGNAL (interperr, $"Unknown
             selection type");                                       0257
          % initialize the selection count i %                       0258
            i ← statesav.nselects+1;                                 0259
          % save away entity type in the state record %              0260
            statesav.entype ← type;                                  0261
            statesav.maxselect ← bugs;                               0262
          % get the reqired number of selections %                   0263
            LOOP                                                     0264
               BEGIN                                                 0265
               % set tptr to point to the result text
               pointer for the selection and save away the
               prompt string %                                       0266
                  IF i = 1                                           0267
                  THEN                                               0268
                     BEGIN                                           0269
                     statesav.cfl1len ← cflstr.L;                    0270
                     &tptr ←  &resultptr;                            0271
                     &promptsav ← &statesav +
                     $selstatesize;                                  0272
                     promptsav.M ← 10; promptsav.L ← 0;
                                                                     0273
                     *promptsav* ← *prompstr*;                       0274
                     END                                             0275
                  ELSE                                               0276
                     BEGIN                                           0277
                     statesav.cfl2len ← cflstr.L;                    0278
                     &tptr ← &resultptr + 2;                         0279
                     END;                                            0280
               % prompt the user if appropriate:  The
               prompt string was put together by evaluate
               after calling sleuth too look ahead in the
               grammar.  It was aslo output there if a
               branching decision had to be made at that
               point. %                                              0281
                  IF inprompts # noprompts AND NOT cueflg
```

```
                                                                    0282
      THEN fbctl( incues, $prompstr );                              0283
% set the processing function according to
what is typed next %                                                0284
   CASE char ← lookc() OF                                           0285
      = cacnar, = inschar, = rptchar:                               0286
        &function ← &bugaction;                                     0287
      = optchar:% address selection %                               0291
        BEGIN                                                       0292
        inpt();                                                     0293
        IF ((curptr.curnodeptr).opcode =
        $dsel) AND (type = $edge) THEN                              02567
           BEGIN                                                    02568
           fbctl( '? );                                             02569
           REPEAT CASE;                                             02570
           END                                                      02571
        ELSE &function ← &oaction;                                  0294
        END;                                                        0295
      = '?:          % request for help %                         0296
        BEGIN                                                       0297
        inpt();                                                     0298
        cueflg ← FALSE;                                             0299
        *helpstr* ← "Please specify a ",
        *(type)*, " by
           ", *(optionstr)*, EOL, "or
        <CTRL-Q> for HELP, or <CTRL-S> for
        SYNTAX", EOL;                                               0300
        fbctl( IF caflag THEN typecalit
        ELSE typelit, $helpstr );                                   0301
        REPEAT CASE;                                                0302
        END;                                                        0303
      = 'S-1COB:  % <↑S> request for SYNTAX
      %                                                             03353
        BEGIN                                                       03354
        inpt();                                                     03355
        cueflg ← FALSE;                                             03356
        cshelp(
        $pathstk+pathx-$totalrecsize,
        cmdmode, FALSE);                                            03361
        REPEAT CASE;                                                03359
        END;                                                        03360
      = CD:                                                         0304
        BEGIN                                                       0305
        inpt();                                                     0306
        SIGNAL (cmdelete);                                          0307
        END;                                                        0308
      = EC,                                                         0309
      = BW:                                                         0310
        BEGIN                                                       0311
        inpt();                                                     0312
        SIGNAL (popstate);                                          0313
        END;                                                        0314
      ENDCASE                                                       0315
        &function ← &defaction;                                     0316
% save away the function address in the
state record %                                                      0317
```

```
   IF i = 1                                          0318
      THEN statesav.sel1fun ← &function              0319
      ELSE statesav.sel2fun ← &function;             0320
% Scall the processing function or put out a
diagnostic %                                         0321
   CASE &function OF                                 0322
      = $getbug,                                     0323
      = $getwindow:                                  0324
        function( parsemode, &tptr );                0325
      = $getdae:                                     0326
        BEGIN                                        0327
        IF i = 1 THEN                                0328
           BEGIN                                     0330
           % initialize text ptr for first
           selection to the current marker
           for the current display area %
                                                     0331
           &ltda ← lda();                            0332
           tptr ← ltda.dacsp;                        0333
           tptr[1] ← ltda.daccnt;                    0334
           END                                       0335
        ELSE                                         0336
           BEGIN                                     0337
           % initialize the text pointer
           for the second address to be the
           result obtained in the first
           selection %                               0338
           tptr ← [&tptr - 2];                       0339
           tptr[1] ← [&tptr - 1];                    0340
           END;                                      0341
        function( parsemode, &tptr );                0342
        END;                                         0343
      = $getlit, = $getid, = $getidlist:             0344
        BEGIN                                        0345
        IF i # 1 THEN REPEAT CASE (0);               0346
        % save a count of the actual number
        of selections obtained %                     0347
           statesav.nselects ← i;                    0348
        function( parsemode, &resultptr,
        &resultptr+2 );                              0349
        EXIT LOOP;                                   0350
        END;                                         0351
      = 0:  % error %                                0352
        BEGIN                                        0353
        inpt();                                      0354
        fbctl( '? );                                 0355
        REPEAT LOOP;                                 0356
        END;                                         0357
   ENDCASE;                                          0358
% save a count of the actual number of
selections obtained %                                0359
   statesav.nselects ← i;                            0360
% set up for another selection if required %
                                                     0361
   IF i >= bugs THEN EXIT LOOP;                      0362
   BUMP i;                                           0363
```

```
                cueflg ← FALSE;                                    0364
                fbctl( echostr, $"through" );                      0365
                arm();                                             01548
             END;                                                  0366
          % invoke the delimiter routine to delimit the
          selection %                                              0367
             CASE &function OF                                     0368
                = $getlit, = $getid, = $getidlist:                 0369
                   BEGIN                                           0370
                   % set tptr to point to second delimiter
                   in string %                                     0371
                      &tptr ← &resultptr + d2sel;                  0372
                   % edit delimiters onto typed in file
                   names or links if not already given %           0373
                      CASE type OF                                 0374
                         = $oldfilelink, = $newfilelink:           0375
                            littolnk( TRUE, &resultptr,
                            &tptr);                                02126
                         = $link:                                  0363
                            littolnk( FALSE, &resultptr,
                            &tptr);                                02127
                         ENDCASE;                                  0391
                   END;                                            0392
                ENDCASE                                            0393
                   IF &delimiter # 0 THEN                          0394
                      BEGIN                                        03144
                      CASE i OF                                    0395
                         = 1: % single selection %                 0396
                            delimiter(&resultptr,
                            &resultptr, &resultptr+2 );            0397
                         = 2: % double selection %                 0398
                         delimiter(&resultptr, &resultptr+2,
                         &resultptr, &resultptr+2);                0399
                         ENDCASE;                                  0400
                      END;                                         03145
          END;                                                     0401
       = backup,     % FALSE parse backup %                       0402
       = cleanup:    % TRUE parsing termination %                 0403
          BEGIN                                                    0404
          FOR i ← statesav.nselects DOWN UNTIL < 1 DO              0405
             BEGIN                                                 0406
             &function ←                                           0407
                IF i = 1                                           0408
                   THEN statesav.sel1fun                           0409
                   ELSE statesav.sel2fun;                          0410
             function(parsemode, &resultptr+2*(i-1) );             0411
             END;                                                  0412
          ctrlchar ← FALSE;                                        01644
          END;                                                     0413
       = popselect:          % pop selection %                     0414
          BEGIN                                                    0415
          slink ← cdlnk ← clpsw ← FALSE;                           02228
          % set up the full actual argument list %                 0416
             &curptr ← &resultptr - $pathrecsize;                  0417
             temp ← statesav.entype;                               0418
          % delete the last selection %                            0419
```

```
                    i ← statesav.nselects;                                  0420
                    IF i = 1                                                0421
                       THEN                                                 0422
                          BEGIN                                             0423
                          cflstr.L ← statesav.cfl1len;                      0424
                          &function ← statesav.sel1fun;                     0425
                          END                                               0426
                       ELSE                                                 0427
                          BEGIN                                             0428
                          cflstr.L ← statesav.cfl2len;                      0429
                          &function ← statesav.sel2fun;                     0430
                          END;                                              0431
                    function(cleanup, &resultptr+2*(i-1) );                 0432
                 % clean up the feedback %                                  0433
                    &promptsav ← &statesav + $selstatesize;                 0434
                    *prompstr* ← *promptsav*;                               0435
                    fbctl( fbpcp );                                         0436
                 % decrement the selection count by 1 %                     0437
                    statesav.nselects ← i-1;                                0438
                 % invoke the selection processor recursively to
                 gather another selection %                                 0439
                    xselect( &resultptr, parsing, $temp, &curptr);
                                                                            0440
                 END;                                                       0441
              ENDCASE;                                                      0442
              RETURN (&resultptr);                                          0443
           END.
                                                                            0444
        (getlit) PROCEDURE( % get a typed literal string %                  02142
           % FORMAL ARGUMENTS %                                             02143
              parsemode,       % parsing mode %                            02144
              tptr1,  % address of starting tptr %                        02145
              tptr2);          % address of finishing tptr %               02146
           % NORMAL RETURNS %                                              02147
              % none %                                                     02148
           % ABNORMAL RETURNS %                                            02528
              % a "popstate" SIGNAL is generated if BC or BW is typed
              and the literal string is empty %                             02529
              % a "statesig" SIGNAL is generated if getstring is out
              of storage space %                                            02530
           LOCAL % VARIABLES %                                            02151
              selstateptr,     % ptr to selstate record %                  02152
              strptr;          % ptr to literal collection string %
                                                                            02153
           REF % VARIABLES %                                              02154
              strptr,                                                       02155
              selstateptr,                                                  02156
              tptr1,                                                        02157
              tptr2;                                                        02158
           % ---------------- %                                            02159
           % set up ptr to the selection state record %                    02160
              &selstateptr ← &tptr1 + pselLen;                              02161
           CASE parsemode OF                                              02162
              = parsing:                                                    02163
                 BEGIN                                                      02164
                 % allocate a literal collection string %                  02531
```

```
      selstateptr.litptr ← &strptr ← 0;                          02532
      ON SIGNAL ELSE                                             02533
         BEGIN                                                   02534
         dismes(1,$"Fatal Storage Error: Please Reset");
                                                                 02535
         clrbuf(FALSE);  % clear input buffers%                  02536
         SIGNAL(statesig);                                       02537
         END;                                                    02538
      &strptr ← getstring( 2000, $dspblk );                      02539
      ON SIGNAL ELSE;                                            02540
   % save ptr to allocated string in the selstate record
   %                                                             02168
      selstateptr.litptr ← &strptr;                              02169
   litreset ← FALSE;                                             02170
   % do prompting for text %                                     02180
      fbctl( incues, $"T:");                                     02181
   % look for a null literal %                                   02171
      IF lookc() = #ctln THEN                                    02172
         BEGIN                                                   02173
         inpt();                                                 02174
         FIND SF(*strptr*) ↑tptr1 ↑tptr2;                        02175
         selstateptr.dolitreset ← FALSE;                         02176
         needconfirm ← TRUE;                                     02177
         RETURN;                                                 02178
         END;                                                    02179
   % read the literal %                                         02182
      IF nlmode = fulldisplay                                    02183
         THEN selstateptr.dolitreset ← TRUE                      02184
         ELSE                                                    02185
            BEGIN                                                02186
            echolt();                                            02187
            selstateptr.dolitreset ← FALSE;                      02188
            END;                                                 02189
      ON SIGNAL ELSE litapflag ← TRUE;                           02552
      setlit();                                                  02190
      CASE rdlit( &strptr, 0) OF                                 02191
         = 3: % BC or BW typed and string already empty
         %                                                       02192
            SIGNAL (popstate);                                   02193
         ENDCASE;                                                02194
      ON SIGNAL ELSE;                                            02553
      IF nlmode = fulldisplay                                    02195
         THEN                                                    02196
            BEGIN                                                02197
            IF littakedown THEN                                  02198
               BEGIN                                             02199
               rstlit();                                         02200
               selstateptr.dolitreset ← FALSE;                   02201
               END;                                              02202
            END                                                  02203
         ELSE echoff();                                          02204
      % set the text ptr to point to the string %                02205
      FIND SF(*strptr*) ↑tptr1 SE(*strptr*) ↑tptr2;              02206
   END;                                                          02207
= backup,                                                        02208
= cleanup;                                                       02209
```

```
            BEGIN                                                   02210
            &selstateptr ← &tptr1 + 4;                              02211
            &strptr ← selstateptr.litptr;                           02212
            IF &strptr THEN freestring( &strptr, $dspblk);          02213
            IF selstateptr.dclitreset                               02214
               AND littakedown                                      02215
               THEN rstlit(); % reset literal feedback area %
                                                                    02216
            END;                                                    02217
      ENDCASE;                                                      02218
      RETURN;                                                       02219
      END.
                                                                    02220
   (getid) PROCEDURE( % get a typed ident string %                  02763
      % FORMAL ARGUMENTS %                                          02764
         parsemode,       % parsing mode %                         02765
         tptr1,  % address of starting tptr %                      02766
         tptr2);          % address of finishing tptr %            02767
      % NORMAL RETURNS %                                           02768
         % none %                                                  02769
      % ABNORMAL RETURNS %                                         02770
         % a "popstate" SIGNAL is generated if BC or BW is typed
         and the literal string is empty %                          02771
         % a "statesig" SIGNAL is generated if getstring is out
         of storage space %                                         02772
      LOCAL % VARIABLES %                                           02773
         selstateptr,     % ptr to selstate record %               02774
         strptr;.         % ptr to literal collection string %
                                                                    02775
      REF % VARIABLES %                                             02776
         strptr,                                                    02777
         selstateptr,                                               02778
         tptr1,                                                     02779
         tptr2;                                                     02780
      % ---------------- %                                         02781
      % set up ptr to the selection state record %                 02782
         &selstateptr ← &tptr1 + psellen;                           02783
      CASE parsemode OF                                             02784
         = parsing:                                                 02785
            BEGIN                                                   02786
            % allocate a literal collection string %               02787
               selstateptr.litptr ← &strptr ← 0;                    02788
               ON SIGNAL ELSE                                       02789
                  BEGIN                                             02790
                  dismes(1,$"Fatal Storage Error: Please Reset");
                                                                    02791
                  clrbuf(FALSE);  % clear input buffers%            02792
                  SIGNAL(statesig);                                 02793
                  END;                                              02794
               &strptr ← getstring( 2000, $dspblk );                02795
               ON SIGNAL ELSE;                                      02796
            % save ptr to allocated string in the selstate record
            %                                                       02797
               selstateptr.litptr ← &strptr;                        02798
            LOOP                                                    02945
               BEGIN                                                02946
```

```
litreset ← FALSE;                                                    02799
% do prompting for text %                                           02800
   fbctl( incues, $"T:");                                           02801
% look for a null literal %                                         02802
   IF lookc() = $ctln THEN                                          02803
      BEGIN                                                         02804
      inpt();                                                       02805
      (nullid):                                                     03276
      FIND SF(*strptr*) ↑tptr1 ↑tptr2;                              02806
      selstateptr.dolitreset ← FALSE;                               02807
      needconfirm ← TRUE;                                           02808
      RETURN;                                                       02809
      END;                                                          02810
% read the literal %                                                02811
   IF nlmode = fulldisplay                                          02812
      THEN selstateptr.dolitreset ← TRUE                            02813
      ELSE                                                          02814
         BEGIN                                                      02815
         echolt();                                                  02816
         selstateptr.dolitreset ← FALSE;                            02817
         END;                                                       02818
   ON SIGNAL ELSE litapflag ← TRUE;                                 02819
   setlit();                                                        02820
   CASE getlid(&strptr) OF                                          03265
      = 1: %got a good ident%                                       03266
      = 2: %query failed, try again%                                03268
         BEGIN                                                      03271
         *strptr* ← NULL;                                           03273
         fbctl(fbaddlit, $"
         Please type another IDENT: ");                             03352
         REPEAT LOOP;                                               03274
         END;                                                       03272
      = 3: %back up the command%                                    03269
         SIGNAL(popstate);                                          03275
      = 4: %null ident%                                             03270
         BEGIN                                                      03279
         rstlit();                                                  03278
         *strptr* ← NULL;                                           03281
         GOTO nullid;                                               03277
         END;                                                       03280
      ENDCASE err($"Undefined return from getlid
      detected in getid");                                          03267
   IF nlmode = fulldisplay                                          02826
      THEN                                                          02827
         BEGIN                                                      02828
         IF littakedown THEN                                        02829
            BEGIN                                                   02830
            rstlit();                                               02831
            selstateptr.dolitreset ← FALSE;                         02832
            END;                                                    02833
         END                                                        02834
      ELSE echoff();                                                02835
% set the text ptr to point to the string %                         02836
   FIND SF(*strptr*) ↑tptr1 SE(*strptr*) ↑tptr2;
                                                                    02837
EXIT LOOP;                                                          02948
```

```
            END;                                                    02847
         END;                                                       02838
      = backup,                                                     02839
      = cleanup:                                                    02840
         BEGIN                                                      02841
         &selstateptr ← &tptr1 + 4;                                 02842
         &strptr ← selstateptr.litptr;                              02843
         IF &strptr THEN freestring( &strptr, $dspblk);             02844
         IF selstateptr.dclitreset                                  02845
            AND littakedown                                         02846
            THEN rstlit(); % reset literal feedback area %
                                                                    02847
         END;                                                       02848
   ENDCASE;                                                         02849
   RETURN;                                                          02850
   END.
                                                                    02851
(getlid) PROCEDURE( % get a typed ident string and do ident
query if requested -- subordinate to GETID and GETIDLIST %
                                                                    03158
   % FORMAL ARGUMENTS %                                            03159
    idstr);  %string for collecting ident%                          03160
   % RETURNS %                                                     03163
      % 1, with idstr having an ident in it %                       03164
      % 2, with idstr null -- query failed %                       03259
      % 3, BC or BW with string empty %                            03258
      % 4, no ident collected -- end of ident list %               03263
   LOCAL i;                                                         03331
   REF % VARIABLES %                                               03171
      idstr;                                                        03175
   % ---------------- %                                            03176
   % read the literal %                                            03208
      CASE rdlit( &idstr, $idntdel) OF                              03218
         = 3: % BC or BW typed and string already empty %           03219
            RETURN(3); %back space old string%                      03220
         ENDCASE;                                                   03221
      LOOP                                                          03351
         IF idstr.L THEN                                            03223
            IF identquery(&idstr) THEN RETURN(1) %successful,
            ident in idstr%                                         03224
            ELSE %user query was unsucessful%                       03262
               BEGIN                                                03225
               IF idstr.L THEN REPEAT LOOP;                         03226
               RETURN(2); %query failed -- no new ident
               specified%                                           03228
               END                                                  03229
         ELSE RETURN(4); %end of ident list or null ident
         collected%                                                 03261
   END.
                                                                    03257
(getidlist) PROCEDURE( % get a typed ident list string %            02852
   % FORMAL ARGUMENTS %                                            03024
      parsemode,      % parsing mode %                             03025
      tptr1,  % address of starting tptr %                        03026
      tptr2);         % address of finishing tptr %               03027
   % NORMAL RETURNS %                                             03028
```

```
   % none %                                                          03029
% ABNORMAL RETURNS %                                                03030
   % a "popstate" SIGNAL is generated if BC or BW is typed
   and the literal string is empty %                                 03031
   % a "statesig" SIGNAL is generated if getstring is out
   of storage space %                                                03032
LOCAL % VARIABLES %                                                  03033
   selstateptr,     % ptr to selstate record %                       03034
   strptr;          % ptr to literal collection string %
                                                                     03035
LOCAL TEXT POINTER z1, z2;                                           03332
LOCAL STRING
   work[50];  %for ident queries%                                    03139
REF % VARIABLES %                                                    03036
   strptr,                                                           03037
   selstateptr,                                                      03038
   tptr1,                                                            03039
   tptr2;                                                            03040
% --------------- %                                                 03041
% set up ptr to the selection state record %                        03042
   &selstateptr ← &tptr1 + psellen;                                  03043
CASE parsemode OF                                                    03044
   = parsing:                                                        03045
      BEGIN                                                          03046
      % allocate a literal collection string %                       03047
         selstateptr.litptr ← &strptr ← 0;                           03048
         ON SIGNAL ELSE                                              03049
            BEGIN                                                    03050
            dismes(1,$"Fatal Storage Error: Please Reset");
                                                                     03051
            clrbuf(FALSE);  % clear input buffers%                   03052
            SIGNAL(statesig);                                        03053
            END;                                                     03054
         &strptr ← getstring( 2000, $dspblk );                       03055
         ON SIGNAL ELSE;                                             03056
      % save ptr to allocated string in the selstate record
      %                                                              03057
         selstateptr.litptr ← &strptr;                               03058
      litreset ← FALSE;                                              03061
      % do prompting for text %                                      03062
         fbctl( incues, $"T:");                                      03063
      % look for a null literal %                                    03064
         IF lookc() = sctln THEN                                     03065
            BEGIN                                                    03066
            inpt();                                                  03067
            FIND SF(*strptr*) ↑tptr1 ↑tptr2;                         03068
            selstateptr.dolitreset ← FALSE;                          03069
            needconfirm ← TRUE;                                      03070
            RETURN;                                                  03071
            END;                                                     03072
      % read the literal %                                           03073
         IF nlmode = fulldisplay                                     03074
            THEN selstateptr.dolitreset ← TRUE                       03075
            ELSE                                                     03076
               BEGIN                                                 03077
               echolt();                                             03078
```

```
                selstateptr.dolitreset ← FALSE;                    03079
                END;                                               03080
            setlit();                                              03082
            DO                                                     03119
               BEGIN                                               03120
               ON SIGNAL ELSE litapflag ← TRUE;                    03081
               CASE rdlit( &strptr, 0) OF                          03083
                  = 3: % EC or BW typed and string already
                  empty %                                          03084
                     SIGNAL (popstate);                            03085
                  ENDCASE;                                         03086
               ON SIGNAL ELSE;                                     03087
               FIND SF(*strptr*) ↑z2;                              03307
               WHILE (FIND z2 ['"/'./] < CH > ↑z1) DO              03301
                  BEGIN                                            03302
                  IF FIND '" THEN FIND ['"/ENDCHR] ↑z2             03304
                  ELSE FIND CH $(LD / '' / SP / '-) $'. ↑z2;
                                                                   03308
                  *work* ← z1 z2;                                  03306
                  IF work.L THEN                                   03086
                     BEGIN                                         03311
                     %                                             03335
                     IF nlmode = fulldisplay THEN rstlit()
                                                                   03315
                     ELSE crlf();                                  03320
                     %                                             03336
                     IF identquery($work) THEN %user query was
                     sucessful%                                    03089
                        ST z1 z2 ← *work*                          03090
                     ELSE                                          03310
                        BEGIN                                      03328
                        FIND z2 > $(SP/',) ↑z2;                    03334
                        ST z1 z2 ← NULL;                           03327
                        END;                                       03329
                     FIND z1 ↑z2;                                  03333
                     END;                                          03312
                  END;                                             03303
               IF *strptr*[strptr.L] NOT= SP THEN                  03131
                  *strptr* ← *strptr*, SP;                         03133
               END                                                 03121
            UNTIL curchr = CA;                                     03122
            WHILE *strptr*[strptr.L] = SP DO                       03134
               BUMP DOWN strptr.L;                                 03135
            IF nlmode = fulldisplay                                03091
               THEN                                                03092
                  BEGIN                                            03093
                  IF littakedown THEN                              03094
                     BEGIN                                         03095
                     rstlit();                                     03096
                     selstateptr.dolitreset ← FALSE;               03097
                     END;                                          03098
                  END                                              03099
               ELSE echoff();                                      03100
         % set the text ptr to point to the string %               03101
            FIND SF(*strptr*) ↑tptr1 SE(*strptr*) ↑tptr2;          03102
         END;                                                      03105
```

```
            = backup,                                                   03106
            = cleanup:                                                  03107
               BEGIN                                                    03108
               &selstateptr ← &tptr1 + 4;                               03109
               &strptr ← selstateptr.litptr;                            03110
               IF &strptr THEN freestring( &strptr, @aspblk);           03111
               IF selstateptr.dclitreset                                03112
                  AND littakedown                                       03113
                  THEN rstlit(); % reset literal feedback area %
                                                                        03114
               END;                                                     03115
            ENDCASE;                                                    03116
         RETURN;                                                        03117
         END.
                                                                        03118
      (identquery) PROCEDURE(idstr); %process Ident Queries%            02950
         LOCAL type;                                                    02951
         LOCAL TEXT POINTER tptr1, tptr2;                               02952
         LOCAL STRING work[500];                                        02953
         REF idstr;                                                     02954
         IF idstr.L = empty THEN RETURN(TRUE);                          02955
         CASE *idstr*[1] OF                                             02956
            ='.: %lastname search%                                      02957
               BEGIN                                                    02958
               *idstr* ← *idstr*[2 TO idstr.L];                         02964
               CCPOS SE(*idstr*);  %check for incomplete name%          02965
               IF READC = '. THEN                                       02966
                  BEGIN %partially specified last name, remove
                  trailing periods%                                     02967
                  DO BUMP DOWN idstr.L UNTIL READC NOT= '.;             02969
                  type ← idchr;                                         02970
                  END                                                   02971
               ELSE type ← lname;                                       02972
               stnamcap(&idstr);                                        02973
               IF NOT idfno THEN idfno ← loaidfil();                    03136
               RETURN( idflsearch(type, &idstr, idfno) );               02974
               END;                                                     02975
            ='": %search name fields of individuals, groups and
            orgs%                                                       02976
               BEGIN                                                    02977
               FIND SF(*idstr*) CH ↑tptr1 ( ['"] ↑tptr2 ←tptr2 /
               SE(*idstr*) ↑tptr2);                                     02983
               *idstr* ← tptr1 tptr2;                                   02984
               IF NOT idfno THEN idfno ← loaidfil();                    03137
               RETURN( idflsearch(strinname, &idstr, idfno) );          02985
               END;                                                     02986
            = SP: %deblank front of string%                             02994
               BEGIN                                                    02995
               *idstr* ← *idstr*[2 TO idstr.L];                         02996
               REPEAT CASE;                                             02997
               END;                                                     02998
            ENDCASE;                                                    02999
         RETURN(TRUE);                                                  03022
         END.
                                                                        03023
      (getbug) PROCEDURE( % get a typed literal string %                 0516
```

```
    % FORMAL ARGUMENTS %                                                0517
       parsemode,       % parsing mode %                                0518
       tptr);  % address of result tptr %                              0519
    % NORMAL RETURNS %                                                  0520
       % none %                                                         0521
    % ABNORMAL RETURNS %                                               0522
       % a "popstate" SIGNAL is generated if BC or BW is typed
       and the literal string is empty %                               0523
    REF % VARIABLES %                                                  0524
       tptr;                                                           0525
    % --------------- %                                                0526
    CASE parsemode OF                                                  0527
       = parsing:                                                      0528
          BEGIN                                                        0529
          needconfirm ← TRUE;                                          0530
          INPUT BUG tptr;                                              0531
          disarm();                                                    0532
          END;                                                         0533
       = backup,                                                       0534
       = cleanup:                                                      0535
          BEGIN                                                        0536
          delbm(); % delete last bug mark %                            0537
          disarm();                                                    0538
          END;                                                         0539
    ENDCASE;                                                           0540
    RETURN;                                                            0541
    END.
                                                                       0542
(getdae) PROCEDURE( % get a typed literal string %                     0543
    % FORMAL ARGUMENTS %                                               0544
       parsemode,       % parsing mode %                                0545
       tptr);  % address of result tptr %                              0546
    % NORMAL RETURNS %                                                  0547
       % none %                                                         0548
    % ABNORMAL RETURNS %                                               0549
       % a "popstate" SIGNAL is generated if BC or BW is typed
       and the literal string is empty %                               0550
    REF % VARIABLES %                                                  0551
       tptr;                                                           0552
    % --------------- %                                                0553
    CASE parsemode OF                                                  0554
       = parsing:                                                      0555
          getaddr(&tptr);                                              0556
       = backup,                                                       0557
       = cleanup:                                                      0558
          IF nlmode = fulldisplay                                      0559
             THEN dn( $""); % clear name buffer %                      0560
    ENDCASE;                                                           0561
    RETURN;                                                            0562
    END.
                                                                       0563
(getwindow) PROCEDURE( % get a window selection %                      0564
    % FORMAL ARGUMENTS %                                               0565
       parsemode,       % parsing mode %                                0566
       tptr);  % address of result tptr %                              0567
    % NORMAL RETURNS %                                                  0568
```

```
    % none %                                                    0569
  LOCAL % variables %                                           0570
    da,     % ptr to display rea %                             0571
    cordsl,         % display window coords %                  0572
    x,      % x coordinate %                                   0573
    y,      % y coordinate %                                   0574
    r;      % division remainder %                             0575
  REF % VARIABLES %                                             0576
    tptr, da;                                                   0577
  % ---------------- %                                         0578
  CASE parsemode OF                                             0579
    = parsing:                                                  0580
      BEGIN                                                     0581
      % check for proper mode and read over the ca
      character %                                               0582
        IF nlmode = typewriter                                  0583
          OR inpt() # cachar THEN                               0584
          err( $"Illegal use of getwindow routine");            0585
      % get the display area address, x and y screen
      coordinates %                                             0586
        IF NOT &da ← dsparea(lcda(:cordsl)) THEN                0587
          err($"lcda failed, getwindow");                       0588
        % X coordinate %                                      02549
          x ← cordsl.xcord - da.daleft;                         0589
          DIV x / da.dahinc, x, r;                              0590
          IF r AND r >= (da.dahinc/2) THEN BUMP x;              0591
          x ← x * da.dahinc;                                    0592
          IF x >= da.daright - da.daleft THEN                  01647
            x ← da.daright - da.daleft - da.dahinc;            02546
              %otherwise MARKIT will fail%                     01649
          cordsl.xcord ← x + da.daleft;                         0593
        % y coordinate %                                      02550
          y ← cordsl.ycord - da.datop;                          0594
          DIV y / da.davinc, y, r;                              0595
          IF r AND r >= (da.davinc/2) THEN BUMP y;              0596
          y ← y * da.davinc;                                    0597
          IF y >= da.dabottom - da.datop THEN                  01650
            y ← da.dabottom - da.datop - da.davinc;            02543
              %otherwise MARKIT will fail%                     01652
          cordsl.ycord ← y + da.datop;                          0598
      %put bug mark on screen%                                  0599
        markit(&da, x, y, SP);                                  0600
      % stuff display area address and translated screen
      coordinates into the result tptr %                        0601
        tptr ← &da;                                             0602
        tptr[1] ← cordsl;                                       0603
      % set the needconfirm flag so we won't back up over
      the confirmation %                                        0604
        needconfirm ← TRUE;                                     0605
      END;                                                      0606
    = backup,                                                   0607
    = cleanup:                                                  0608
      % delete the bug mark from the screen %                   0609
        delbm();                                                0610
  ENDCASE;                                                      0611
  RETURN;                                                       0612
```

```
        END.                                                            0613
    % DELIMITER ROUTINES %                                              0614
      (stdr) PROCEDURE( % truncates a bug selection to a statement
      selection %                                                       0615
        bug,        % text ptr for bug %                             0616
        tptr1,% first text pointer %                                    0617
        tptr2);% second text pointer %                                  0618
        REF bug, tptr1, tptr2;                                          0619
        %---------------%                                              0620
        FIND SF(bug) ↑tptr1 SE(bug) ↑tptr2;                             0621
        RETURN;                                                         0622
        END.
                                                                        0623
      (grpdr) PROCEDURE( % delimits a structural group %                0624
        % FORMAL ARGUMENTS %                                           0625
          tptr1,  % addrss of first text pointer %                      0626
          tptr2,  % address of second text pointer %                    0627
          ret1,   % address of the first result text pointer % 0628
          ret2);  % address of the second result text pointer %
                                                                        0629
        % RETURNS %                                                    0630
          % text pointers ret1 and ret2 are set up to delimit the
          group identified by the bug selections tptr1 and tptr2 %
                                                                        0631
        REF % VARIABLES %                                               0632
          tptr1, tptr2, ret1, ret2;                                     0633
        %-------------%                                                0634
        ret1[1] ← ret2[1] ← 1;                                          0635
        ret1 ← grptst( tptr1, tptr2: ret2);                             0636
        RETURN;                                                         0637
        END.
                                                                        0638
      (plxdr) PROCEDURE( % delimits a structural plex %                 0639
        % FORMAL ARGUMENTS %                                           0640
          tptr1,  % addrss of first text pointer %                      0641
          ret1,   % address of the first result text pointer % 0642
          ret2);  % address of the second result text pointer %
                                                                        0643
        % RETURNS %                                                    0644
          % text pointers ret1 and ret2 are set up to delimit the
          plex identified by the bug selection tptr1 %                  0645
        REF % VARIABLES %                                               0646
          tptr1, ret1, ret2;                                            0647
        %-------------%                                                0648
        ret1[1] ← ret2[1] ← 1;                                          0649
        ret1 ← plxset( tptr1: ret2);                                    0650
        RETURN;                                                         0651
        END.
                                                                        0652
    % LEVELADJ STRING %                                                0653
      (xlevadj) PROCEDURE( % collect leveladj string %                02419
        % FORMAL ARGUMENTS %                                           02420
          resultptr,      % ptr to the return argument record %
                                                                        02421
          parsemode);% parsing mode %                                   02422
```

```
% NORMAL RETURNS %                                                    02423
   % 1) returns the leveladj count (d = -1, u = +1 etc ) %
                                                                       02424
   % 2) returns the leveladj string %                                 02425
% ABNORMAL RETURNS %                                                  02426
LOCAL % VARIABLES %                                                   02427
   char,    % inpt char %                                             02428
   levadjstr;       % collection string for leveladj %                02429
REF % VARIABLES %                                                     02430
   levadjstr,                                                         02431
   resultptr;                                                         02432
REF tda, inpt, fbstr, sysmsg;                                         03338
%-----------------%                                                   02433
% set up a string descriptor to point to the level adjust
collection string %                                                   02434
   &levadjstr ← &resultptr + plvlen;                                  02435
% process according to parsemode %                                   02436
   CASE parsemode OF                                                  02437
      = parsing:                                                      02438
         BEGIN                                                        02439
         % initialize the count to 0 %                               02440
            resultptr ← 0;                                            02441
         % initialize string lengths %                               02442
            levadjstr.M ← 25;                                         02443
            levadjstr.L ← 0;                                          02444
         % if no level adjust then just exit %                       02445
            IF nolevadj THEN RETURN(&resultptr);                      02446
         % prompt the user for a level adj %                         02447
            fbctl( incues, $"L:" );                                   02448
         % collect the level adjust string %                         02449
            LOOP                                                      02450
               BEGIN                                                  02451
               IF nlmode = fulldisplay                                02452
                  THEN dn( &levadjstr );                              02453
               CASE char ← inpt() OF                                  02454
                  =SP, =cachar, =inschar, =rptchar: %
                  terminators %                                       02455
                     EXIT LOOP;                                       02456
                  ='u:                                                02457
                     BEGIN                                            02458
                     BUMP resultptr;                                  02459
                     *levadjstr* ← *levadjstr*, char;                 02460
                     IF nlmode # fulldisplay                          02461
                        THEN typech( char );                          02462
                     END;                                             02463
                  ='d:                                                02464
                     BEGIN                                            02465
                     BUMP DOWN resultptr;                             02466
                     *levadjstr* ← *levadjstr*, char;                 02467
                     IF nlmode # fulldisplay                          02468
                        THEN typech( char );                          02469
                     END;                                             02470
                  ='?:                                                02471
                     fbctl( typelit, $"Please type a
                     LEVEL-ADJUST string:
                           u / d / <SPACE> / <CA>
```

```
                or <CTRL-Q> for HELP, or <CTRL-S> for
                SYNTAX");                                   02472
             = 'S-100B:  % <↑S> request for SYNATX %
                                                           03362
                cshelp( $pathstk+pathx-$totalrecsize,
                cmdmode, FALSE);                           03363
             =BC:                                          02473
                IF levadjstr.L > 0                         02474
                   THEN                                    02475
                      BEGIN                                02476
                      CASE *levadjstr*/levadjstr.L/ OF
                                                           03342
                         = 'd: BUMP resultptr;             03343
                         = 'u: BUMP DOWN resultptr;
                                                           03344
                         ENDCASE;                          03345
                      IF nlmode # fulldisplay THEN
                                                           02477
                         BEGIN                             02478
                         typech(BC);                       02479
                         typech( *levadjstr*/
                         levadjstr.L / );                  02480
                         END;                              02481
                      levadjstr.L ← levadjstr.L - 1;
                                                           02482
                      END                                  02483
                   ELSE                                    02484
                      SIGNAL(popstate);                    02485
             =BW:                                          02486
                IF levadjstr.L > 0                         02487
                   THEN                                    02488
                      BEGIN                                02489
                      IF nlmode # fulldisplay              02490
                         THEN typech(BW);                  02491
                      *levadjstr* ← NULL;                  02492
                      resultptr ← 0;                       03339
                      END                                  02493
                   ELSE                                    02494
                      SIGNAL(popstate);                    02495
             =CD:                                          02496
                SIGNAL (cmdelete);                         02497
          ENDCASE                                          02498
             BEGIN                                         02499
             % return the processed chars to the inpt
             buffer %                                      02500
                unput();                                   02501
                IF levadjstr.L > 0 THEN resetb(
                &levadjstr );                              02502
             % clear result string %                       02503
                *levadjstr* ← NULL;                        02504
             % reset the count %                           02505
                resultptr ← 0;                             02506
             % clear name area IF DNLS %                   02507
                IF nlmode = fulldisplay                    02508
                   THEN dn( &levadjstr );                  02509
             EXIT LOOP;                                    02510
```

```
                        END;                                                02511
                     END;                                                   02512
                  IF nlmode = typewriter THEN                               02513
                     BEGIN                                                  02514
                     crlf();                                                02515
                     END;                                                   02516
                  END;                                                      02517
               = backup,                                                    02518
               = cleanup:                                                   02519
                  BEGIN                                                     03346
                  resultptr ← 0;                                            03347
                  IF nlmode = fulldisplay THEN                              02520
                     BEGIN                                                  02521
                     *levadjstr* ← NULL;                                    02522
                     dn( &levadjstr );                                      02523
                     END;                                                   02524
                  END;                                                      03348
            ENDCASE;                                                        02525
         RETURN (&resultptr);                                               02526
         END.
                                                                            02527
   % VIEWSPEC STRING %                                                       0761
      (xviewspecs) PROCEDURE( % processes viewspecs %                       02260
         % the current viewspecs are modified by the current inpt
         string.  when a delimiter for the viewspec string is
         encountered, the updated viewspecs are returned.  Note that
         the new viewspecs are not saved in the display area
         viewspec words %                                                   02261
         % FORMAL ARGUMENTS %                                              02262
            resultptr,      % ptr to the return argument record %
                                                                            02263
            parsemode);% parsing mode %                                     02264
         % NORMAL RETURNS %                                                02265
            % 1) pvsrecord %                                               02266
            % 2)... viewspec collection string %                           02267
         % ABNORMAL RETURNS %                                              02268
         LOCAL % VARIABLES %                                                02269
            da,     % ptr to display area %                                02270
            vspec[2],       % viewspec words %                             02271
            save,   % flag save word %                                      02272
            sav1,   % viewspec save word for recreate display %             02273
            sav2,   % viewspec save word for recreate display %             02274
            vstr,   % collection string for viewspecs %                     02275
            goodvs,         % flag set for good viewspecs %                 02276
            char;   % current inpt char %                                   02277
         REF % VARIABLES %                                                  02278
            da,                                                             02279
            vstr,                                                           02280
            resultptr;                                                      02281
         REF tda, inpt, fbstr, sysmsg;                                      02562
         %----------------%                                                 02282
         % set vstr to point to the viewspec collection string %
                                                                            02283
            &vstr ← &resultptr + pvslen;                                    02284
         % process according to parsemode %                                02285
            CASE parsemode OF                                               02286
```

```
= parsing:                                                    02287
  BEGIN                                                       02288
  % initialize the viewspec collection string %               02289
    vstr.L ← 0; vstr.M ← 20;                                  02290
  % set da to point to the current display area %
                                                              02291
    &da ← lda();                                              02292
  % save user's display area progs in the vsrecord %
                                                              02293
    resultptr.vscacode ← da.dacacode;                         02294
    resultptr.vsusqcod ← da.dausqcod;                         02295
  % if no viewspecs then just return current ones %
                                                              02296
    IF novspec THEN                                           02297
      BEGIN                                                   02298
      resultptr ← da.davspec;                                 02299
      resultptr[1] ← da.davspc2;                              02300
      RETURN(&resultptr);                                     02301
      END;                                                    02302
  % save the current viewspecs in the viewspec save
  area %                                                      02303
    vspec ← IF cspupdate THEN cspvs ELSE
    da.davspec;                                               02304
    vspec[1] ← IF cspupdate THEN cspvs[1] ELSE
    da.davspc2;                                               02305
  % prompt the user for viewspecs %                          02306
    fbctl( incues, $"V:");                                    02307
  % set up for viewspec collection %                         02308
    RESET savevspec;                                          02309
    IF nlmode = fulldisplay THEN                              02310
      BEGIN                                                   02311
      dspvsp( vspec, vspec[1], 3); % display them
      %                                                       02312
      ltl();                                                  02313
      af();                                                   02314
      END                                                     02315
    ELSE                                                      02316
      BEGIN                                                   02317
      IF buffs # buffn THEN typebuff();                       02318
      echolt();                                               02319
      END;                                                    02320
  % collect the viewspec string %                            02321
  LOOP                                                        02322
    BEGIN                                                     02323
    CASE char ← inpt() OF                                     02324
      =cachar, =inschar, =rptchar: % terminators %
                                                              02325
        BEGIN                                                 02326
        % set return viewspec words %                         02327
          resultptr ← vspec;                                  02328
          resultptr[1] ← vspec[1];                            02329
        % clean up the feedback %                            02330
          IF nlmode = fulldisplay THEN                        02331
            BEGIN                                             02332
            lts();                                            02333
            an ();                                            02334
```

```
            END                                          02336
         ELSE echoff();                                  02337
      EXIT LOOP;                                         02338
      END;                                               02339
=CD:                                                     02340
      BEGIN                                              02341
      % clean up the feedback %                          02342
         IF nlmode = fulldisplay THEN                    02343
            BEGIN                                        02344
            dspvsp( da.davspec, da.davspc2, 3);
                                                         02563
            lts();                                       02345
            an ();                                       02346
            END                                          02348
         ELSE echoff();                                  02349
      SIGNAL (cmdelete);                                 02350
      END;                                               02351
='f:                                                     02352
      IF nlmode = fulldisplay                            02353
         THEN                                            02354
            BEGIN                                        02355
            IF vspec.vsrlev THEN                         02356
               vspec ← <SEQGEN, reslev>(vspec,
               getlev(da.dacsp));                        02357
            save ← da.davspec.vsdaft := TRUE;
                                                         02358
            sav1 ← da.davspec := vspec;                  02359
            sav2 ← da.davspc2 := vspec[1];               02360
            dafrmt (&da, 0);                             02361
            da.davspec.vsdaft ← save;                    02362
            da.davspec ← sav1;                           02363
            da.davspc2 ← sav2;                           02364
            END;                                         02365
=BC:                                                     02366
      IF vstr.L > 0                                      02367
         THEN                                            02368
            BEGIN                                        02369
            IF nlmode # fulldisplay THEN                 02370
               BEGIN % $echo deleted chars %
                                                         02371
               typech( BC );                             02372
               typech( *vstr* [vstr.L] );                02373
               END;                                      02374
            bkc( &vstr );                                02375
            POP savevspec TO vspec;                      02376
            END                                          02377
         ELSE                                            02378
            BEGIN                                        02379
            SIGNAL (popstate);                           02380
            END;                                         02381
='?:                                                     02382
      fbctl( typelit, $"Please type in
      VIEWSPECS or <CTRL-Q> for HELP or
      <CTRL-S> for SYNTAX");                             02383
= 'S-100B:  % <↑S> request for SYNATX %                  03364
      cshelp( $pathstk+pathx-$totalrecsize,
```

```
                        cmdmode, FALSE);                                03365
                    ENDCASE                                             02384
                        BEGIN                                           02385
                        % the current viewpsec words (vspec) are
                        pushed onto the savevspec stack and setlt is
                        called to put the current viewspec into
                        effect %                                        02386
                            PUSH vspec ON savevspec;                    02387
                            vspec ← setlt (char, vspec, vspec[1] :
                            vspec[1], goodvs);                          02388
                            IF NOT goodvs THEN dismes(2, $"Undefined
                            viewspec -- ignored");                      02389
                        % append the current character to the
                        viewspec string %                               02390
                            *vstr* ← *vstr*, char;                      02391
                        END;                                            02392
                    % the viewspec display is updated if in DNLS %
                                                                        02393
                        IF nlmode = fulldisplay THEN                    02394
                            BEGIN                                       02395
                            dn($vstr);                                  02396
                            dspvsp (vspec, vspec[1], 3);                02397
                            END;                                        02398
                    END; % of LOOP %                                    02399
                                                                        02400
                    END;                                                02401
                = backup,                                               02402
                = cleanup:                                              02403
                    BEGIN                                               02404
                    % if no viewspecs then just exit %                  02405
                        IF novspec THEN EXIT CASE;                      02406
                    % clean up the feedback %                           02407
                        IF nlmode = fulldisplay THEN                    02408
                            BEGIN                                       02409
                            % cleanup the display %                     02564
                                &da ← lda();                            02565
                                dspvsp( da.davspec, da.davspc2, 3);     02566
                            lts();                                      02410
                            an ();                                      02411
                            dn ($"");                                   02412
                            END                                         02413
                        ELSE echoff();                                  02414
                        END;                                            02415
            ENDCASE;                                                    02416
        RETURN (&resultptr);                                            02417
        END.
                                                                        02418
% COMMAND CONFIRMATION RECOGNIZER %                                      0908
    (xconfirm) PROCEDURE( % process confirmation character %            02574
        % FORMAL ARGUMENTS %                                            02575
        resultptr,      % ptr to the return argument record %
                                                                        02576
        parsemode,      % parsing mode %                                02577
        lastopcode);% opcode of previous interpreter inst. %
                                                                        02578
        % NORMAL RETURNS %                                              02579
```

```
    % 1) returns the command completion code %                    02580
    % the current command completion code is saved in the
    global complcode and has the following values %               02581
       % = 1: normal command completion %                         02582
       % = 2: insert statement mode %                             02583
       % = 3: repeat command mode %                               02584
% ABNORMAL RETURNS %                                             02585
REF % VARIABLES %                                                02586
   resultptr;                                                    02587
   REF tda, inpt, fbstr, sysmsg;                                 02588
%-----------------%                                              02589
% process according to parsemode %                              02590
   CASE parsemode OF                                             02591
      = parsing:                                                 02592
         BEGIN                                                   02593
         % check to see in the last inst was a $call on a
         built in recognition function.  If it was, then we
         back up the inpt buffer by 1 character and use the
         terminator character of that recognizer as the
         inpt character for the confirmation character
         recognizer %                                            02594
            IF lastopcode IN [$ssel, $levadj]                    02595
               AND buffs = buffn % inpt buffer is empty %
                                                                 02596
               AND NOT needconfirm                               02597
               THEN unput()                                      02598
               ELSE fbctl(incues, $"OK:" );                      02599
         % process the next character, looking for a
         terminator %                                            02600
            CASE inpt() OF                                       02601
               = rptchar:                                        02602
                  complcode ← 3;                                 02603
               = inschar:                                        02604
                  complcode ← 2;                                 02605
               = cachar:                                         02606
                  NULL;                                          02607
               = BC, = BW:                                       02608
                  SIGNAL (popstate);                             02609
               = CD:                                             02610
                  SIGNAL (cmdelete);                             02611
               = '?:                                             02612
                  BEGIN                                          02613
                  fbctl( typelit, $"Please type an OK
                  character or <CTRL-Q> for HELP or
                  <CTRL-S> for SYNTAX");                         02614
                  REPEAT CASE;                                   02615
                  END;                                           02616
               = 'S-100B:  % <↑S> request for SYNATX %           03366
                  BEGIN                                          03367
                  cshelp( $pathstk+pathx-$totalrecsize,
                  cmdmode, FALSE);                               03368
                  REPEAT CASE;                                   03369
                  END;                                           03370
            ENDCASE                                              02617
               BEGIN                                             02618
               fbctl( '? );                                      02619
```

```
                        REPEAT CASE;                                    02620
                        END;                                            02621
                    resultptr ← ccmplcode;                              02622
                    END;                                                02623
            ENDCASE;                                                    02624
        RETURN (&resultptr);                                            02625
        END.                                                            02626

% entity finding routines - known as delimiters %                        0961
    (cdr) PROCEDURE (bug,ptr1,ptr2);                                    0962
        REF bug, ptr1, ptr2;                                            0963
        FIND bug > ↑ptr1 CH ↑ptr2;                                      0964
        RETURN;                                                         0965
        END.                                                            0966
                                                                        0967
    (idr) PROCEDURE (bug,ptr1,ptr2);                                    02554
        LOCAL TEXT POINTER tp1;                                         02555
        REF bug, ptr1, ptr2;                                            02556
        FIND bug ↑tp1 < $NP ↑ptr1 tp1 > CH $NP ↑ptr2;                   02557
        RETURN;                                                         02558
        END.                                                            02559
                                                                        02560
    (tdr) PROCEDURE (bug1,bug2,ptr1,ptr2);                              01998
        REF bug1, bug2, ptr1, ptr2;                                     01999
        LOCAL TEXT POINTER tp1;                                         02000
        IF POS SF(bug1) # SF(bug2) THEN err($"invalid text selection")
                                                                        02001
        ELSE                                                            02002
            IF POS bug1 < bug2 THEN                                     02003
                FIND bug1 ↑ptr1 bug2 > CH ↑ptr2                         02004
            ELSE                                                        02005
                FIND bug2 ↑tp1 bug1 > CH ↑ptr2 tp1 ↑ptr1;               02006
        RETURN;                                                         02007
        END.                                                            02008
                                                                        02009
    (ndr) PROCEDURE (bug, ptr1, ptr2); %verify number %                  0965
        LOCAL sflg;                                                     0986
        REF bug, ptr1,ptr2;                                             0987
        sflg ← TRUE;                                                    0988
        IF NOT FIND bug < ($(D/ '-/ '+/ '-/ '+) ↑ptr1)                  0989
        THEN err($"invalid number selection");                          0990
        IF NOT FIND ptr1 > ('-/'+/FR sflg) ((1$3D $(', 3D) (', 1$D/-D)
        ↑ptr2)/                                                         0991
            (1$D (', 1$D/TRUE) ↑ptr2)/(', 1$D ↑ptr2))                   0992
        THEN err($"invalid number selection");                          0993
        FIND ptr2 < -D ↑ptr2;                                           0994
        RETURN;                                                         0995
        END.                                                            0996
                                                                        0997
    (wdr) PROCEDURE (bug,ptr1,ptr2);
        % This procedure delimits a "WORD" given a pointer to a
        character in a string.  The pointers PTR1 and PTR2 are set to
        the first and last characters, respectively, of the word.%
                                                                        02542
        REF bug, ptr1, ptr2;                                            0998
        FIND bug > CH $LD ↑ptr2 bug < $LD ↑ptr1;                        0999
```

```
   RETURN;                                                          01000
   END.                                                             01001
                                                                    01002
(nmdr) PROCEDURE (bug,ptr1,ptr2);                                   01003
   % statement name (and statement number) delimiter %              01004
   REF bug, ptr1, ptr2;                                             01005
   FIND bug >                                                       01006
      $(LD / '- /'' / '@  ) ↑ptr2                                   01007
      bug < $(LD / '- /'' / '@  ) ↑ptr1;                            01008
      % @ is alphabetic zero %                                      01009
   RETURN;                                                          01010
   END.                                                             01011
                                                                    01012
(iddr) PROCEDURE (bug,ptr1,ptr2); %IDENT delimiter routine%         02640
   REF bug, ptr1, ptr2;                                             02642
   FIND bug >                                                       02759
       $1('&/'↑) $(LD / '- /'' ) ↑ptr2                              02762
      bug < $(LD / '- /'' ) ↑ptr1;                                  02761
   RETURN;                                                          02647
   END.                                                             02648
                                                                    02649
(idldr) PROCEDURE (bug1,bug2,ptr1,ptr2);                            02661
   REF bug1, bug2, ptr1, ptr2;                                      02662
   LOCAL TEXT POINTER tp1, tp2;                                     02663
   IF POS SF(bug1) # SF(bug2) THEN err($"invalid IDENTLIST
   selection -- must be contained in one statement")                02664
   ELSE                                                             02665
      IF POS bug1 < bug2 THEN                                       02666
         FIND bug1 ↑ptr1 bug2 > CH ↑ptr2                            02667
      ELSE                                                          02668
         FIND bug2 ↑tp1 bug1 > CH ↑ptr2 tp1 ↑ptr1;                  02669
   iddr(&bug1, &ptr1, $tp2);                                        02670
   iddr(&bug2, &tp1, &ptr2);                                        02679
   RETURN;                                                          02670
   END.                                                             02671
                                                                    02672
(vdr) PROCEDURE (bug,ptr1,ptr2);                                    01013
   REF bug, ptr1, ptr2;                                             01014
   FIND bug > CH $PT ↑ptr2 bug < $PT ↑ptr1;                         01015
   RETURN;                                                          01016
   END.                                                             01017
                                                                    01018
(flndr) PROCEDURE (bug,ptr1,ptr2); %file name -- TENEX syntax%
                                                                    01019
   REF bug, ptr1, ptr2;                                             01020
   FIND bug > $(LD/'./';/'>/'-) ↑ptr2 bug < $(LD/'./';/'>/'-/'<)
   ↑ptr1;                                                           01021
   RETURN;                                                          01022
   END.                                                             01023
                                                                    01024
(ldr) PROCEDURE (bug,ptr1,ptr2);                                    02130
   REF bug, ptr1, ptr2;                                             02132
   FIND bug ↑ptr1 ↑ptr2;                                            02139
   RETURN;                                                          02136
   END.                                                             02137
                                                                    02138
```

```
%test and set bounds of structure%                                      01033
                                                                        01034
   (plxset)
      %Given a stid, this routine will return the stid's of the head
      and tail, respectively, of the plex in which the stid
      appears.%                                                         01035
      %--------------%                                                  01036
      PROCEDURE(stid);                                                  01037
      LOCAL grp1, %stid of head%                                        01038
         grp2; %stid of tail%                                           01039
      IF stid.stpsid = orgstid THEN grp1 ← grp2 ← stid                  01040
      ELSE                                                              01041
         BEGIN                                                          01042
         grp1 ← gethed(stid);                                           01043
         grp2 ← getail(stid);                                           01044
         END;                                                           01045
      RETURN(grp1, grp2);                                               01046
      END.
                                                                        01047
% DAE PARSER %                                                         01128
   (getaddr) PROCEDURE(ptr); %read and evaluate address expression%
                                                                        01841
      %parameter: address of tpointer--starting tpointer for            01842
      relative addresses                                                01843
      Returns: after having updated the tpointer                        01844
      Error conditions: displays error message if receives unusual
      inpt                                                              01845
      %                                                                 01846
      %----------------%                                               01847
      REF ptr;                                                          01849
      LOCAL nklink;                                                     01850
      LOCAL TEXT POINTER oldptr, z1, z2;                                01851
      LOCAL STRING                                                      01852
         gadstr[200]; % to hold the entire string for the address %
                                                                        01853
                                                                        02140
      (gad):                                                            01854
      ON SIGNAL ELSE IF nlmode = fulldisplay THEN rstlit();             01855
      slink ← cdlnk ← TRUE;                                             02224
      *gadstr* ← NULL;                                                  01856
      gaddrlit($gadstr); %collect the literal%                          01857
      FIND SF(*gadstr*) ↑z1 SE(*gadstr*) ↑z2;                           02124
      littolnk( FALSE, $z1, $z2);                                       02125
      oldptr ← ptr;                                                     01894
      oldptr[1] ← pt°[1];                                               01895
      FIND SF(*gadstr*) ↑z1 SE(*gadstr*) ↑z2;                           01896
      ON SIGNAL %catch any errors during interpretation%                01897
         =gaderr:                                                       01898
            BEGIN                                                       01899
            *[MESSAGE]*←SP,*[MESSAGE]*,"?  ";                           01900
            dismes(2, MESSAGE);                                         01901
            &sysmsg ← $"";                                              01902
            cueflg ← FALSE;                                             01903
            clrbuf(0);                                                  01904
            ptr ← oldptr;                                               01905
            ptr[1] ← oldptr[1];                                         01906
```

```
            GOTO gad;                                                   01907
            END;                                                        01908
         ELSE;                                                          01909
      cspvs ← caddexp($z1, $z2, lda(), &ptr: cspvs[1]); %interpret
      the string%                                                       01910
      RETURN END.
                                                                        01911
   (gaddrlit) PROCEDURE( % get a typed in address expression %          01552
      % FORMAL ARGUMENTS %                                              01553
         strptr);   % address of collection string %                   01556
      % NORMAL RETURNS %                                                01557
         % none %                                                       01558
      % ABNORMAL RETURNS %                                              01559
         % a "popstate" SIGNAL is generated if BC or BW is typed and
         the literal string is empty %                                  01560
      REF % VARIABLES %                                                 01564
         strptr;                                                        01567
      % ---------------- %                                             01569
      litreset ← FALSE;                                                 01577
      % do prompting for text %                                         01583
         fbct1( incues, $"A:");                                         01584
      % look for a null literal %                                       01575
         IF lookc() = $ctln THEN                                        01576
            BEGIN                                                       01627
            inpt();                                                     01626
            RETURN;                                                     01629
            END;                                                        01628
      % read the literal %                                             01585
         IF nlmode = typewriter THEN                                    01586
            echolt();                                                   01590
         CASE rdlit( &strptr, 0) OF                                     01593
            = 3: % BC or BW typed and string already empty %            01594
               SIGNAL (popstate);                                       01595
            ENDCASE;                                                    01596
         IF nlmode = fulldisplay AND littakedown THEN rstlit()          01597
         ELSE echoff();                                                 01606
      RETURN;                                                           01621
      END.
                                                                        01622
   (littolnk)        % turn a lit into a link maybe %                   02010
      PROCEDURE                                                         02011
         (type,      % TRUE -> should be old/new filelink %            02012
         tp1,        % address of text pointer to start of lit %       02013
         tp2);       % address of text pointer to end of lit %         02014
      LOCAL temp;                                                       02128
      REF tp1, tp2;                                                     02015
      temp ← type;                                                      02063
      FIND SF(tp1) $(SP/TAB);                                           02064
      IF FIND ( '( / '< / "--" ) THEN temp ← TRUE;                      02065
      LOOP                                                              02066
         CASE READC OF                                                  02067
            = '), = '>, = ',, = '::                                     02068
               BEGIN                                                    02069
               temp ← TRUE;                                             02070
               END;                                                     02071
            = '':                                                       02072
```

```
            CASE READC OF                                                   02073
               = ENDCHR:                                                    02074
                  err($"Missing Character After Single Quote");
                                                                            02075
               ENDCASE;                                                     02076
         = '"':                                                             02077
            LOOP                                                            02078
               CASE READC OF                                                02079
                  = '": EXIT LOOP;                                          02080
                  = ENDCHR:                                                 02081
                     BEGIN                                                  02082
                     ST tp1 ← SF(tp1) SE(tp1), '";                          02083
                     tp2[1] ← tp2[1] + 1;                                   02097
                     EXIT LOOP 2;                                           02084
                     END;                                                   02085
                  ENDCASE;                                                  02086
         = ENDCHR:  EXIT LOOP;                                              02087
         ENDCASE;                                                           02088
      IF temp OR type THEN                                                  02089
         BEGIN                                                              02090
         IF NOT FIND SF(tp1) > $(SP/TAB) ( '( / '< / "--" ) THEN
                                                                            02091
            BEGIN                                                           02098
            ST tp1 ← '<, SF(tp1) SE(tp1);                                   02092
            tp2[1] ← tp2[1] + 1;                                            02100
            END;                                                            02099
         FIND SE(tp1) $(SP/TAB);                                            02105
         CASE type OF                                                       02102
            = TRUE:                                                         02103
               CASE READC OF                                                02112
                  = '), ='>: NULL;                                          02113
                  = ',:                                                     02114
                     BEGIN                                                  02115
                     ST tp1 ← SF(tp1) SE(tp1), '>;                          02116
                     tp2[1] ← tp2[1] + 1;                                   02117
                     END;                                                   02118
               ENDCASE                                                      02119
                  BEGIN                                                     02120
                  ST tp1 ← SF(tp1) SE(tp1), ",>";                           02121
                  tp2[1] ← tp2[1] + 2;                                      02122
                  END;                                                      02123
            ENDCASE                                                         02104
               CASE READC OF                                                02106
                  = '), ='>: NULL;                                          02107
                  ENDCASE                                                   02108
                     BEGIN                                                  02109
                     ST tp1 ← SF(tp1) SE(tp1), '>;                          02094
                     tp2[1] ← tp2[1] + 1;                                   02111
                     END;                                                   02110
         END;                                                               02095
      RETURN;                                                               02129
      END.
                                                                            02096
FINISH                                                                      01536
```

SEQFIL

(MLK)SEQFIL (MLK)SEQFIL (MLK)SEQFIL (MLK)SEQFIL (MLK)SEQFIL (M
(MLK)SEQFIL (MLK)SEQFIL (MLK)SEQFIL (MLK)SEQFIL (MLK)SEQFIL (M

```
< NLS, SEQFIL.NLS;181, >, 15-OCT-74 09:11 KJM ;;;;< NLS,
SEQFIL.NLS;171, >, 11-JUN-74 18:19 KEV ;

FILE seqfil % L10 <REL-NLS>Seqfil %% (L10,) (rel-nls,Seqfil.rel,) %
                                                                    02

%....declarations.....%
                                                                    03
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4, p = 7, wa = 8;          04
   REF prseqwk, tda;                                                05
   DECLARE                                                          06
      maxlev = 63, n40fst = 40B, tenfst = 0,                        07
      sfhfnm = 48, sfhpgn = 70,                                     08
      seqer = 4B8, seqeof = 1B9, maxtrc = 63, maxmclv = 1,          09
      ascmbk = 175B;
                                                                    010
   DECLARE                                                          011
      ok=1,          % successful return %                         012
      cat=0,         % catastrophic error return %                 013
      submission=1, % journal submission %                         014
      leaving=-1,    % file leaving system %                       015
      entering=0;    % file entering system %                      016
   DECLARE levind = (0,0,0,0,0,0,0,0,0,0,0,0);                     017
   DECLARE STRING formfeed = "
```

```
    ";                                                                  018
%.....output sequential, quickprint, and assembler.....%
                                                                        019
    (outseq) PROCEDURE   % does the Output Sequential File command %   020
       (ofilnam,   % string containing name of the output file %        021
       da,         % display area descriptor %                          022
       forceup);   % whether alphas are to be forced upper case %       023
       REF ofilnam, da;                                                 024
       dismes(1, $"Output Sequential in Progress");                     025
       opseqf (&ofilnam, &da, forceup, FALSE, opsqfl);                  026
       dismes(0);                                                       027
       RETURN;                                                          028
       END.
                                                                        029
    (outqp)  PROCEDURE   % does the Output Quickprint command %        030
       (ofilnam,   % string containing name of the output file %        031
       da);        % display area descriptor %                         032
       REF ofilnam, da;                                                 033
       dismes (1, $"Output Quickprint in Progress");                    034
       opseqf (&ofilnam, &da, FALSE, TRUE, opqpfl);                     035
       dismes(0);                                                       036
       RETURN;                                                          037
       END.
                                                                        038
    (outjm) PROCEDURE ( str, da ); % does Output Journal Mail command %
                                                                        039
       % FORMAL PARAMETERS %                                           040
          % str  - address of astring containing output file name %    041
          % address of display area %                                  042
       LOCAL pjb, pjba, pjbb, pjbc;                                  02073
          %may delete this statement and make pjb a formal parameter%
                                                                     02095
       LOCAL pjl, pjbrad, pjfile;                                    02094
       LOCAL pjbflag %journal branch found%, pjoflag %mail found%;   02081
       LOCAL pjcsp, pjrubmrk, pjcaccde;                                 043
       LOCAL TEXT POINTER pjtp, tp1, tp2;                               044
       REF str, da, pjb;                                                045
       % set branch names, etc. %                                    02074
          pjbflag ← pjoflag ← 0;                                     02082
          %may delete these when pjb is made a calling parameter%    02096
             &pjb ← $pjba;                                           02075
             pjba←$"Journal"; pjbb←$"Info"; pjbc←$"Action";          02076
          pjl←3;                                                     02077
          pjfile ← da.dacsp.stfile;                                  02093
       WHILE (pjl←pjl-1) >= 0 DO IF (pjbrad ← pjb[pjl]) THEN         02078
          BEGIN                                                      02079
          % fetch stid for head of Journal branch %                     046
             bl ← origin; % build stid %                                047
             bl.stfile ← pjfile; % for origin statemnt %                048
             bl[1] ← 1; % of currently loaded file %                    049
             specreg (pjbrad, nametyp, $bl); % fetch stid for Journal,
             Action, or Info branch %                                   050
             IF bl = endfil THEN REPEAT LOOP; % there may not be any %
                                                                        051
             pjbflag ← TRUE;                                         02098
             IF getsub(bl) = bl THEN REPEAT LOOP; %no mail here%     02099
```

```
      pjtp ← bl; % save stid for head %                                052
      pjtp[1] ← bl[1]; % of Journal branch %                           053
   IF NOT pjoflag THEN                                              02085
      BEGIN                                                         02083
      % save away current display area csp and viewspecs %          054
        pjusqc ← da.dausqcod; % save user seq gen addr %            055
        pjsavf ← da.davspec; % and viewspecs %                      056
        pjcsp ← da.dacsp; % save csp for restoration %              057
        pjrubmrk ← rubmrk; % save pointer to curr rubout flag %
                                                                    058
        pjcacode ← da.dacacode;                                     059
      % trap signals so can retore the dispay area stuff %         060
        ON SIGNAL ELSE % can't afford to lose control %             061
          BEGIN                                                     062
          da.davspec ← pjsavf; % w/o restore user seq gen addr
          %                                                         063
          da.dausqcod ← pjusqc; % and viewspecs %                   064
          da.dacsp ← pjcsp; % restore csp %                         065
          da.dacacode ← pjcacode;                                   066
          rubmrk ← pjrubmrk; % restore ruboout flag pointer %       067
          END;                                                      068
      % set up display area stuff for new sequence %               069
        da.dausqcod ← $pjseqg; % instate Print Journal seq gen %
                                                                    070
        da.davspec.vsusqf ← TRUE; % and enable it %                 071
        da.davspec.vsbrof ← TRUE; % branch only %                   072
        da.dacacode ← $pager; % content analyzer to put out form
        feeds %                                                     074
        da.davspec.vscapf ← TRUE; % turn on content analyzer %
                                                                    075
        rubmrk ← $pjrubout; % intercept RUBOUT %                    076
      % set flag so this won't be done again %                   02086
        pjoflag ← TRUE;                                         02087
      dismes(1, $"Output of Journal Mail in progress" );            078
      END;                                                      02084
    % output the file %                                            077
      da.dacsp ← bl;                                                073
      opseqf( &str, &da, FALSE, TRUE, opqpfl );                     079
   END;                                                         02080
IF NOT pjbflag THEN err($" No Journal Branches");               02097
IF pjoflag THEN                                                 02088
   BEGIN                                                        02089
   % restore the saved display area state stuff %                  081
      dismes(0);                                                    080
      da.davspec ← pjsavf; % restore user seq gen addr %            082
      da.dausqcod ← pjusqc; % and  viewspecs %                      083
      da.dacsp ← pjcsp; % restore csp %                             084
      da.dacacode ← pjcacode;                                       085
      rubmrk ← pjrubmrk; % restore rubout flag pointer %            086
   END                                                          02090
ELSE err($" You have no Journal mail");                         02092
RETURN; % return to caller %                                      087
END.
                                                                  088
(pager) PROCEDURE ( sw );                                         089
   % FORMAL PARAMETERS %                                          090
```

```
      % sw   address of the sequence generator work area %                091
   % TEXT POINTERS %                                                      092
      LOCAL TEXT POINTER  tpl;   % local copy of CCPOS %                  093
   REF sw;                                                                094
   % set tpl to be the current CCPOS -- do not use CCPOS as it might
   be used by some routine downstream from us %                           095
      FIND ↑tpl;                                                          096
   % look for a statment that looks like a journal citation %             097
      IF FIND tpl /"Location:"/ THEN                                      098
         BEGIN % journal citation, eject to a new page %                  099
         send(&sw,$formfeed);                                            0100
         END;                                                            0101
   RETURN(TRUE);                                                         0102
   END.                                                                  0103
(outasm) PROCEDURE   % does the Output to Assembler File command %      0104
   (ofilnam,   % string containing name of the output file %             0105
   da,         % display area descriptor %                               0106
   forceup);   % whether alphas are to be forced upper case %           0107
   REF ofilnam, da;                                                      0108
   dismes(1, $"Output for Assembler in Progress");                       0109
   opseqf (&ofilnam, &da, forceup, FALSE, opmcfl);                       0110
   dismes(0);                                                            0111
   RETURN;                                                               0112
   END.                                                                  0113
(opseqf) PROCEDURE                                                      01666
   (ofilnam,   % string containing name of the output file %            01667
   da,         % display area descriptor %                              01668
   upflg,      % whether to force alphabetic characters uppercase %
                                                                        01669
   pgflg,      % whether to paginate %                                   01670
   typfil);    % type of file %                                          01671
                                                                        01672
   %This is the main control file for generating a sequential file.
   Basically, it uses the sequence generator to get new statements
   and then adds these statements to a sequential file.  The file is
   both opened and closed here. %                                       01673
   %--------------------%                                              01674
   LOCAL jfn, sw, toplev, seqstd, sv1, sv2;                             01675
   LOCAL STRING string[200];                                            01676
   REF ofilnam, da, sw;                                                 01677
                                                                        01678
   % open the output file %                                            01679
      (opsqfa):                                                         01680
      IF (NOT oqapfg) OR ( FIND SF(*ofilnam*) '< *prtdir* '> ) THEN
                                                                        01681
         toplev ← gtjoosf                                               01784
      ELSE toplev ← 0;                                                  01682
      IF NOT jfn ←                                                      01683
         lgetjfn (0, &ofilnam, $txtext, toplev, $lit) THEN              01684
            SIGNAL (ofilerr, $lit);                                     01685
      !gtfdb( jfn, 1B6+1, $r3);                                         01686
      IF oqapfg THEN oqapfg ← IF r3.fdbnxf THEN FALSE ELSE TRUE;
                                                                        01687
```

```
    IF NOT sysopen (jfn, IF oqapfg THEN append ELSE write, chrtyp,
    $lit) THEN                                                      01688
       BEGIN                                                        01689
       reljfn (jfn);                                                01690
       SIGNAL (ofilerr, $lit);                                      01691
       END;                                                         01692
 %initialize formatting stuff%                                      01693
    sfucf  ← upflg;                                                 01694
    sfpgno ← pgflg;                                                 01695
    sflnel ← 72;                                                    01696
    sfndlv ← da.daind/da.dahinc;                                    01697
    sfmxnd ← da.damind/da.dahinc;                                   01698
 %initialize sequence generator%                                    01699
    &sw ← openseq (seqstd ← da.dacsp, seqend(seqstd, sv1 ←
    da.davspec, sv2 ← da.davspc2), sv1, sv2, da.dausqcod,
    da.dacacode);                                                   01700
 %initialize stuff for doing pmap's%                                01701
    sfpmr1.LH ← 1B5;                                                01702
    sfpmr1.RH ← $sfbuff/1000B;                                      01703
    sfpmr2.LH ← jfn;                                                01704
    sfpmr2.RH ← 0;                                                  01705
    sfpmr3 ← 140001B6;                                              01706
    sfbyte ← 0;                                                     01707
    sfbufl ← slngth(                                                01708
       (sfbptr ← stbptr(empty) + $sfbuff),                          01709
       (sfndbf ← stbptr(5) + $sfbufe) );                            01710
 %initialize for output quickprint append%                          01711
 IF oqapfg AND (typfil = opqpfl) THEN                               01712
    BEGIN                                                           01713
    r1 ← jfn;                                                       01714
    r2 ← 'L - 100B;                                                 01715
    !JSYS bout;                                                     01716
    END;                                                            01717
 IF (IF typfil=opqpfl THEN &sw ELSE sfnxtst(&sw)) THEN              01718
    BEGIN                                                           01719
    IF sfpgno THEN                                                  01720
       BEGIN                                                        01721
       bldsfhdr(&da);                                               01722
       *string* ← CR, LF, *sfhead*, 'l, 22B;                        01723
       sflnpg ← 3;                                                  01724
       CCPOS SF(*string*);                                          02262
       sftrin(stbptr(empty)+$string+1,
       stbptr(string.L)+$string+1);                                 01725
       BUMP sfpgno;                                                 01726
       END;                                                         01727
    CASE typfil OF                                                  02263
       = opmcfl:                                                    02264
          BEGIN                                                     01729
          IF (toplev ← sw.swclvl) = 0 THEN toplev ← 1;              01730
          sfmacpt(&da, &sw, toplev);                                01731
          END;                                                      01732
       = opqpfl: NULL;                                              02267
       ENDCASE sfputst(&da, &sw, typfil);                           02266
    WHILE sfnxtst(&sw) DO                                           01734
       IF typfil=opmcfl THEN sfmacpt(&da, &sw, toplev)              01735
       ELSE sfputst(&da, &sw, typfil);                              01736
```

```
      END;                                                          01737
   % close sequence %                                               01738
      closeseq (&sw);                                               01739
   %close file%                                                     01740
      IF NOT oqapfg THEN                                            01741
         BEGIN                                                      01742
         %pmap last page out%                                       01743
            r1 ← sfpmr1;                                            01744
            r2 ← sfpmr2;                                            01745
            r3 ← sfpmr3;                                            01746
            !JSYS pmap;                                             01747
         END                                                        01748
      ELSE                                                          01749
         BEGIN                                                      01750
         %sout last page out%                                       01751
            r1←sfpmr2.LH;                                           01752
            r2.RH←sfpmr1.RH*1000B;                                  01753
            r2.LH←-1;                                               01754
            r3←0;                                                   01755
            !JSYS sout;                                             01756
         END;                                                       01757
      %dismiss page from fork%                                      01758
         r1 ← -1;                                                   01759
         r2 ← sfpmr1;                                               01760
         r3 ← sfpmr3;                                               01761
         !JSYS pmap;                                                01762
      sfbyte ← sfbyte + slngth((stbptr(empty) + $sfbuff), sfbptr);
                                                                    01763
      r1.LH ← 12B;                                                  01764
      r1.RH ← jfn;                                                  01765
      r2 ← -1;                                                      01766
      r3 ← sfbyte;                                                  01767
      !JSYS chfdb;                                                  01768
      % close output file (but keep jfn) %                         01769
         IF NOT sysclose (jfn .V 4B11, $lit) THEN                   01770
            BEGIN                                                   01771
            reljfn (jfn);                                           01772
            dismes (2, $lit);                                       01773
            GOTO STATE;                                             01774
            END;                                                    01775
      % delete old versions, if not Output Quickprint or Mailfile %
                                                                    01776
         IF typfil # opqpfl AND typfil # opmlfl THEN delovsrns (jfn,
         nvtk);                                                     01777
      % releas the jfn %                                           01778
         reljfn (jfn);                                              01779
   RETURN;                                                          01780
   END.                                                             01781
                                                                    01782
(sfnxtst) PROCEDURE (sw);                                           0233
   %This procedure calls the sequence generator to find the next
   statement.  If there are no more statements in the sequence, it
   returns FALSE.  Otherwise, it initializes CCPOS for reading the
   next statement, which may involve skipping the statement name. %
                                                                    0234
   %-----------------%                                             0235
```

```
   LOCAL TEXT POINTER tptr;                                              0236
   REF sw;                                                               0237
   IF inptrf OR (tptr ← seqgen(@sw)) = endfil THEN                       0238
      RETURN(FALSE);                                                     0239
   tptr[1] ← IF sw.swvspec.vsnamf THEN 1                                 0240
      ELSE fchtxt(getsdb(tptr));                                         0241
   CCPOS tptr;                                                           0242
   RETURN(TRUE);                                                         0243
   END.                                                                  0244
                                                                         0245
(sfputst) PROCEDURE (da, sw, typfil);                                    02100
   %This copies the contents of READC to a local string. It expects
   READC to be set up to begin reading.  When it completes
   processing this statement it performs the necessary cleanup to
   begin processing the next statement (by calling SFENDST) and
   returns.%                                                             02101
   %------------------%                                                  02102
   LOCAL TEXT POINTER tptr;                                              02103
   LOCAL curlin, chrcnt, indcnt, begbp, bytptr, char, tabpos, count,
   inblnk, outblnk, frstblnk, cntrlstr, mrgnlvl, extraline, sigend,
   stnobp, prevchar, string[50];                                         02104
   LOCAL STRING stnsig[30];                                              02105
   REF da, sw;                                                           02106
   %initialize this statement%                                           02107
      IF (sw.swvspec.vsplxf OR sw.swvspec.vsbrof) THEN mrgnlvl ←
      sw.swslvl                                                          02108
         ELSE mrgnlvl ← 1;                                               02109
      bytptr ← begbp ← stbptr(empty) + $string;                          02110
      indcnt ← IF (NOT sw.swvspec.vsindf AND NOT (sw.swvspec.vsbrof
      OR sw.swvspec.vsplxf)) THEN 0                                      02111
         ELSE MAX (0, MIN (sfndlv * (sw.swclvl-mrgnlvl), sfmxnd));
                                                                         02112
      IF (chrcnt ← indcnt) > 0 THEN                                      02113
         FOR count ← 1 UP UNTIL > indcnt DO ↑bytptr ← SP;                02114
      IF sw.swvspec.vsstnf AND NOT sw.swvspec.vsstnr                     02115
      AND sw.swcstid.stpsid # origin THEN                                02116
         BEGIN % statement number on left %                              02117
         *stnsig* ← NULL;                                                02118
         IF sw.swvspec.vssidf                                            02119
            THEN % display sid's %                                       02120
               *stnsig* ← '0, STRING( getsid( sw.swcstid ) )             02121
            ELSE % display line numbers%                                 02122
               fechnm (sw.swsvw, $stnsig);                               02123
         FOR count ← empty + 1 UP UNTIL > stnsig.L DO                    02124
            ↑bytptr ← *stnsig*[count];                                   02125
         ↑bytptr ← SP;                                                   02126
         chrcnt ← chrcnt + stnsig.L + 1;                                 02127
         END;                                                            02128
      curlin ← 1;                                                        02129
      extraline ← FALSE;                                                 02268
   LOOP   % for each line %                                              02130
      BEGIN                                                              02131
      inblnk ← CCPOS;                                                    02132
      outblnk ← frstblnk ← bytptr;                                       02133
      LOOP                                                               02134
         CASE char ← READC OF                                            02135
```

```
= SP:                                                          02136
   BEGIN                                                       02137
   IF (chrcnt ← chrcnt + 1) >= sflnel THEN EXIT LOOP 1;
                                                               02139
   ↑bytptr ← SP;                                               02138
   outblnk ← bytptr;                                           02140
   inclnk ← CCPOS;                                             02141
   prevchar ← CCPOS;                                           02304
   END;                                                        02142
= ENDCHR: EXIT LOOP 2;                                         02143
= TAB:                                                         02144
   BEGIN                                                       02145
   tabpos ← MIN (sflnel,                                       02146
      fndtab(&da,chrcnt + 1) - 1);                             02147
   FOR chrcnt UP UNTIL >= tabpos DO                            02148
      ↑bytptr ← SP;                                            02149
   IF chrcnt > sflnel THEN EXIT LOOP 1;                        02150
   outblnk ← bytptr;                                           02151
   inblnk  ← CCPOS;                                            02152
   END;                                                        02153
= CR, =EOL:  EXIT LOOP 1;                                      02154
< 40B:  %handle control chars for output quickprint%
                                                               02155
   IF typfil NOT= opqpfl THEN GO TO eccntrl                    02156
   ELSE  %put in printing strings for control
   characters%                                                 02157
      BEGIN                                                    02158
                                                               02159
      %force page eject on ↑L & don't print anything%
                                                               02160
      IF (char = 'L - 100B) AND sfpgno THEN                    02161
         BEGIN                                                 02162
         sflnpg ← 200;                                         02163
         GO TO eccntrl;                                        02164
         END;                                                  02165
                                                               02166
      %get printing string and put in output string%
                                                               02167
      cntrlstr ← npstrad(char);                                02168
      IF chrcnt + /cntrlstr/.L <= sflnel THEN                  02298
         BEGIN  %stash away printing string if it fits%
                                                               02299
         FOR count ← empty + 1 UP UNTIL > /cntrlstr/.L
         DO                                                    02169
            BEGIN                                              02170
            ↑bytptr ← */cntrlstr/*/count/;                     02171
            chrcnt ← chrcnt + 1;                               02172
            END;                                               02173
         chrcnt ← chrcnt - 1;                                  02174
         END                                                   02301
      ELSE chrcnt ← chrcnt + /cntrlstr/.L - 1;                 02302
                                                               02175
      %continue as if normal characters%                       02176
      GO TO eccntrl;                                           02177
      END;                                                     02178
ENDCASE                                                        02179
```

```
                BEGIN                                                   02160
                (eccntrl):                                              02164
                   IF (chrcnt ← chrcnt + 1) >= sflnel THEN              02165
                      BEGIN                                             02166
                      IF frstblnk # outblnk THEN %there are blanks%
                                                                        02167
                         BEGIN                                          02168
                         bytptr ← outblnk;                              02169
                         %reset readwk%                                 02190
                            tptr ← sw.swstid;                           02191
                            tptr[1] ← inblnk;                           02192
                            CCPOS tptr;                                 02193
                         END                                            02194
                      ELSE   %no blanks; back up 1 char%                02291
                         BEGIN                                          02292
                         tptr ← sw.swstid;                              02293
                         tptr[1] ← prevchar;                            02294
                         CCPOS tptr;                                    02295
                         END;                                           02296
                      EXIT LOOP 1;                                      02195
                      END;                                              02196
                IF char > 40B THEN ↑bytptr ← IF sfucf AND char IN
                ['a, 'z] THEN                                           02182
                   char - 40B ELSE char;                                02183
                prevchar ← CCPOS;                                       02303
                END;                                                    02197
          IF curlin >= sw.swvspec.vstrnc THEN EXIT LOOP 1;              02198
          ↑bytptr ←  CR;                                                02199
          ↑bytptr ←  LF;                                                02200
          sftrln (begbp, bytptr);   % transfer line to output buffer %
                                                                        02201
          % intialize for next line %                                   02202
             BUMP curlin;                                               02203
             bytptr ← begbp;                                            02204
             IF (chrcnt ← indcnt) > 0 THEN                              02205
                FOR count ← 1 UP UNTIL > indcnt DO ↑bytptr ← SP;         02206
          END;                                                          02207
       % end of statement stuff %                                       02208
       IF sw.swvspec.vsstnf AND sw.swvspec.vsstnr                       02209
       AND sw.swcstid.stpsid # origin THEN                              02210
          BEGIN  %statement number on right%                            02211
          *stnsig* ← NULL;                                              02212
          IF sw.swvspec.vssidf                                          02213
             THEN % display sid's %                                     02214
                *stnsig* ← '0, STRING( getsid( sw.swcstid ) )           02215
             ELSE % display line numbers%                               02216
                fechnm (sw.swsvw, $stnsig);                             02217
          IF chrcnt + 1 + stnsig.L > sflnel THEN                        02218
             BEGIN %must go to new line%                                02219
             ↑bytptr ← CR;                                              02220
             ↑bytptr ← LF;                                              02221
             sftrln (begbp, bytptr);   % transfer line to output buffer
             %                                                          02222
             bytptr ← begbp;                                            02223
             chrcnt ← 0;                                                02224
             extraline ← TRUE;                                          02269
```

```
        END                                                          02225
      ELSE extraline ← FALSE;                                        02270
      FOR count ← chrcnt + 1 UP UNTIL > sflnel - stnsig.L DO         02226
        ↑bytptr ← SP;                                                02227
      FOR count ← empty + 1 UP UNTIL > stnsig.L DO                   02228
        ↑bytptr ← *stnsig*[count];                                   02229
      END;                                                           02230
    IF NOT (sw.swvspec.vsidtf AND extraline) THEN                    02271
      BEGIN          %only if signatures on and go on separate line%
                                                                     02272
      ↑bytptr ← CR;                                                  02231
      ↑bytptr ← LF;                                                  02232
      sftrln (begbp, bytptr);    % transfer line to output buffer %
                                                                     02233
      sigend ← sflnel;                                               02273
      END                                                            02274
    ELSE                                                             02275
      BEGIN                                                          02276
      sigend ← sflnel - stnsig.L - 3;                                02277
      stnobp ← bytptr;                                               02278
      END;                                                           02279
    IF sw.swvspec.vsidtf THEN                                        02234
      BEGIN                                                          02235
      bytptr ← begbp;                                                02236
      *stnsig* ← NULL;                                               02237
      fechsig (sw.swcstid, $stnsig);                                 02238
      FOR count ← 1 UP UNTIL > sigend - stnsig.L DO                  02239
        ↑bytptr ← SP;                                                02240
      FOR count ← empty + 1 UP UNTIL > stnsig.L DO                   02241
        ↑bytptr ← *stnsig*[count];                                   02242
      IF extraline THEN bytptr ← stnobp;                             02261
      ↑bytptr ← CR;                                                  02243
      ↑bytptr ← LF;                                                  02244
      sftrln (begbp, bytptr);    % transfer line to output buffer %
                                                                     02245
      END                                                            02246
    ELSE IF sw.swvspec.vsblkf THEN                                   02247
      BEGIN                                                          02248
      bytptr ← begbp;                                                02249
      ↑bytptr ← CR;                                                  02250
      ↑bytptr ← LF;                                                  02251
      sftrln (begbp, bytptr);    % transfer line to output buffer %
                                                                     02252
      END;                                                           02253
    RETURN;                                                          02254
    END.
                                                                     02255
(sfmacpt) PROCEDURE (da, sw, toplev);                                0416
    %This copies the contents of READC to a local string, for output
    to the assembler. It works like sfputst, except it inserts tabs
    at the beginning of statements that are of a lower level than the
    first statement in the series. In addition a statement consisting
    of only one space will be output as a null line. %               0417
    %--------------------%                                           0418
                                                                     0419
    LOCAL char, bytptr, count, begptr, tmpptr, string[400];          0420
```

```
REF da, sw;                                                          0421
                                                                     0422
% initialization %                                                   0423
begptr ← bytptr ← tmpptr ← stbptr(empty) + $string;                  0424
                                                                     0425
% put out semi-colon in front of origin statement %                 0426
IF (NOT sw.swstid.stastr) AND (sw.swstid.stpsid = origin) THEN
                                                                     0427
   ↑bytptr ← '; ;                                                    0428
                                                                     0429
%  put out leading tabs in front of first line of statement if
appropriate and indenting is turned on  %                            0430
IF (count ← sw.swclvl - toplev) < 0  THEN count ← 0;                 0431
IF count > 0 AND da.davspec.vsindf THEN                              0432
   DO ↑bytptr ← TAB  UNTIL (count ← count - 1) <= 0;                 0433
tmpptr ← bytptr;                                                     0434
                                                                     0435
LOOP                                                                 0436
   BEGIN                                                             0437
   LOOP                                                              0438
      CASE  char ← READC  OF                                         0439
         = ENDCHR:  EXIT LOOP 1;                                     0440
         = CR, =EOL:                                                 0441
            BEGIN                                                    0442
            ↑bytptr ← CR;                                            0443
            ↑bytptr ← LF;                                            0444
            EXIT LOOP 1;                                             0445
            END;                                                     0446
         ENDCASE                                                     0447
            BEGIN                                                    0448
            % convert to upper case if appropriate %                 0449
            ↑bytptr ← IF sfucf AND char IN ['a, 'z] THEN             0450
               char - 40B ELSE char;                                 0451
            END;                                                     0452
                                                                     0453
   % put out null statement if appropriate %                         0454
   IF char = ENDCHR AND ↑tmpptr = SP AND slngth(tmpptr,bytptr) =
   0                                                                 0455
      THEN bytptr ← begptr;                                          0456
                                                                     0457
   CASE char OF                                                      0458
      = ENDCHR: EXIT LOOP 1;                                         0459
      = EOL, = CR: NULL;                                             0460
      ENDCASE                                                        0461
         BEGIN                                                       0462
         ↑bytptr ← CR;                                               0463
         ↑bytptr ← LF;                                               0464
         END;                                                        0465
   sftrln(begptr, bytptr);  % transfer line to output buffer %
                                                                     0466
                                                                     0467
   % reinitialize for going thru loop %                              0468
   begptr ← tmpptr ← bytptr ←stbptr(empty) + $string;                0469
   END;                                                              0470
                                                                     0471
↑bytptr ← CR;                                                        0472
```

```
   ↑bytptr ← LF;                                                        0473
   sftrln(begptr, bytptr);  % transfer line to output buffer %          0474
   RETURN;                                                              0475
   END.                                                                 0476
                                                                        0477
(sftrln) PROCEDURE (begbp, bytptr);                                     0478
   %This routine copies characters bounded by begbp and bytptr (both
   byte pointers) into sfbuff and pmaps sfbuff into the print file
   when the buffer is full.%                                            0479
   %------------------%                                                0480
   LOCAL STRING string[100];                                            0481
   UNTIL begbp = bytptr DO                                              0482
      BEGIN                                                             0483
      ↑sfbptr ← ↑begbp;                                                 0484
      IF sfbptr = sfndbf THEN                                           0485
         BEGIN                                                          0486
         IF NOT oqapfg THEN                                             0487
            BEGIN                                                       0488
            %pmap page out%                                             0489
               r1 ← sfpmr1;                                             0490
               r2 ← sfpmr2;                                             0491
               r3 ← sfpmr3;                                             0492
               !JSYS pmap;                                              0493
            END                                                         0494
         ELSE                                                           0495
            BEGIN                                                       0496
            %sout page out%                                             0497
               r1←sfpmr2.LH;                                            0498
               r2.RH←sfpmr1.RH*1000B;                                   0499
               r2.LH←-1;                                                0500
               r3←-5000B;                                               0501
               !JSYS sout;                                              0502
            END;                                                        0503
         %dismiss page from fork%                                       0504
            r1 ← -1;                                                    0505
            r2 ← sfpmr1;                                                0506
            r3 ← sfpmr3;                                                0507
            !JSYS pmap;                                                 0508
         sfbptr ← stbptr(empty) + $sfbuff;                              0509
         BUMP sfpmr2;                                                   0510
         sfbyte ← sfbyte + sfbufl;                                      0511
         END;                                                           0512
      END;                                                              0513
   IF sfpgno AND (sflnpg ← sflnpg + 1) >= 60 THEN                       0514
      BEGIN                                                             0515
      *string* ← 14S, CR, LF, *sfhead*, STRING(sfpgno), 22B;            0516
      sflnpg ← 3;                                                       0517
      sftrln (stbptr(empty)+$string+1, stbptr(string.L) +
      $string+1);                                                       0518
      BUMP sfpgno;                                                      0519
      END;                                                              0520
   RETURN;                                                              0521
   END.                                                                 0522
                                                                        0523
(bldsfhdr) PROCEDURE (da);                                              0524
   %This routine builds a page header for output quickprint.%           0525
```

```
    %------------------%                                                0526
    LOCAL vspwrd, count;                                                0527
    LOCAL STRING filstr[100];                                           0528
    REF da;                                                             0529
    IF oqnhfg THEN                                                     02256
       BEGIN                                                           02257
       %No header wanted; page numbers only%                           02258
          *sfhead* ← "
                   Page ";            %57 spaces%                     02259
          RETURN;                                                      02260
       END;                                                            02261
    *filstr* ← NULL;                                                    0530
    vspwrd ← da.davspec;                                                0531
    %put initials into sfhead%                                          0532
       *sfhead* ← "    ", *initsr*, ",    ";                          0533
    getdat($sfhead);                                                    0534
    %put truncation and level values into sfhead%                       0535
       IF vspwrd.vstrnc < maxtrc OR vspwrd.vslev < maxlev THEN          0536
          BEGIN                                                         0537
          IF vspwrd.vstrnc < maxtrc THEN                                0538
             *sfhead* ← *sfhead*, " X=", STRING(vspwrd.vstrnc), ',
                                                                        0539
          ELSE *sfhead* ← *sfhead*, " T=ALL,";                          0540
          IF vspwrd.vslev < maxlev THEN                                 0541
             *sfhead* ← *sfhead*, " L=", STRING(vspwrd.vslev), ',       0542
          ELSE *sfhead* ← *sfhead*, " L=ALL,";                          0543
          END;                                                          0544
    %get file name%                                                     0545
       filnam(da.dacsp.stfile, $filstr);                                0546
    %blank fill to position for file name%                              0547
       FOR count ← 1 UP UNTIL > MAX (1, sfhpgn - filstr.L - sfhead.L)
       DO                                                              01783
          *sfhead* ← *sfhead*, SP;                                      0548
    %put file name in header%                                           0549
       *sfhead* ← *sfhead*, *filstr*;                                   0550
    %2 blanks to position for page number%                              0551
       *sfhead* ← *sfhead*, "  ";                                       0552
    RETURN;                                                             0553
    END.                                                                0554
                                                                        0555
%.....insert sequential.....%
                                                                        0556
    (inseq) PROCEDURE   % does the Copy Sequential command %            0557
       (stid,      % statement after which to insert %                  0558
       rlevcnt,    % level relative to stid %                           0559
       ifilnam,    % string containing the name of the input file %     0560
       filtyp);    % type of input %                                    0561
                                                                        0562
       % This is the main control routine for insert sequential.  The
       input file is opened and closed here. %                          0563
       %------------------%                                             0564
       LOCAL count, levblk, lstblk, curblk, curdpth, levchg, char, pos,
       offset, jfn, t1, t2, t3, t4, t5, t6, term, firststid;            0565
       LOCAL TEXT POINTER start, end;                                   0566
       REF ifilnam;                                                     0567
                                                                        0568
```

```
dismes(1, $"Copy Sequential in Progress");                              0569
firststid←0;                                                           02282
% open the input file %                                                 0570
   IF NOT jfn ← lgetjfn (0, &ifilnam, $txtext, gtjoisf, $lit)
   THEN                                                                0571
      SIGNAL (cfilerr, $lit);                                          0572
   IF NOT sysopen (jfn, read, chrtyp, $lit) THEN                       0573
      BEGIN                                                            0574
      reljfn (jfn);                                                    0575
      SIGNAL (cfilerr, $lit);                                          0576
      END;                                                             0577
IF filtyp = assfil THEN GOTO inseqaws;                                 0578
levblk ← lstblk ← curdpth ← 0;                                         0579
offset ← IF filtyp = n4ofil THEN n4ofst ELSE tenfst;                   0580
*lit* ← NULL;                                                          0581
isread (jfn, $sar, CR);                                                0582
pos ← empty + 1;                                                       0583
WHILE *sar*[pos] = LF DO BUMP pos;                                     0584
IF filtyp # macfil THEN                                                0585
   CASE *sar*[pos]  OF                                                 0586
      =ascmbk:                                                         0587
         levblk ← lstblk ← *sar*[pos ← pos + 1] - offset;              0588
      =SP, =TAB:                                                       0589
         BEGIN                                                         0590
         levblk ← 1;                                                   0591
         LOOP                                                          0592
            CASE *sar*[pos ← pos + 1] OF                               0593
               =SP, =TAB: BUMP levblk;                                 0594
               ENDCASE EXIT LOOP;                                      0595
         lstblk ← levblk;                                              0596
         pos ← pos - 1;                                                0597
         END;                                                          0598
      ENDCASE pos ← pos - 1                                            0599
ELSE pos ← pos - 1;                                                    0600
curdpth ← 0;                                                           0601
LOOP                                                                   0602
   BEGIN                                                               0603
   UNTIL (pos := pos + 1) > sar.L DO                                   0604
      CASE char ← *sar*[pos] OF                                        0605
         =ascmbk:                                                      0606
            BEGIN                                                      0607
            count ← *sar*[pos ← pos + 1] - offset;                     0608
            UNTIL (count := count - 1) = 0 DO                          0609
            *lit* ← *lit*, SP;                                         0610
            END;                                                       0611
         = LF: NULL;                                                   0612
         = ENDCHR: IF lit.L NOT= empty THEN GOTO insql1 ELSE GOTO
         insql2;                                                       0613
         = EOL, =CR: %end of statement%                                0614
            BEGIN                                                      0615
            (insql1):                                                  0616
               FIND SF(*lit*) ↑start SE(*lit*) ↑end;                   0617
               stid ← <STRMNP, cinssta> (stid, rlevcnt, $start,
               $end);                                                  0618
               IF firststid=0 THEN firststid←stid;                    02283
            (insql2): *lit* ← NULL;                                    0619
```

```
IF NOT isread(jfn, $sar, CR) THEN EXIT LOOP 2;                    0620
pos ← empty + 1;                                                  0621
WHILE *sar*[pos] = LF DO BUMP pos;                                0622
IF filtyp # macfil THEN                                           0623
   CASE *sar*[pos] OF                                             0624
      %determine level of next statement%                         0625
      =ascmbk:                                                    0626
         curblk ← *sar*[pos ← pos + 1] - offset;                  0627
      =SP, =TAB:                                                  0628
         BEGIN                                                    0629
         count ← 1;                                               0630
         LOOP                                                     0631
            CASE *sar*[pos ← pos + 1] OF                          0632
               =SP, =TAB:                                         0633
                  BEGIN                                           0634
                  IF filtyp = macfil AND count >=
                  maxmclv THEN EXIT LOOP;                         0635
                  BUMP count;                                     0636
                  END;                                            0637
               ENDCASE EXIT LOOP;                                 0638
         pos ← pos - 1;                                           0639
         curblk ← count;                                          0640
         END;                                                     0641
      ENDCASE                                                     0642
         BEGIN                                                    0643
         curblk ← 0;                                              0644
         pos ← pos - 1;                                           0645
         END                                                      0646
ELSE                                                              0647
   BEGIN                                                          0648
   curblk ← 0;                                                    0649
   pos ← pos - 1;                                                 0650
   END;                                                           0651
IF curblk THEN                                                    0652
   BEGIN                                                          0653
   IF NOT levblk THEN levblk ← curblk;                            0654
   IF filtyp # macfil THEN                                        0655
      CASE lstblk OF                                              0656
         =curblk: %same level%                                    0657
            rlevcnt ← levsuc;                                     0658
         < curblk: %levdown a level%                              0659
            IF (curblk =_ lstblk) >= levblk                        0660
            OR curblk/levblk > curdpth THEN                       0661
               BEGIN                                              0662
               rlevcnt ← levdown;                                 0663
               BUMP curdpth;                                      0664
               END                                                0665
            ELSE rlevcnt ← levsuc;                                0666
         ENDCASE %up a level%                                     0667
            BEGIN                                                 0668
            levchg ← (lstblk - curblk) / levblk;                  0669
            IF levchg < 1 AND curdpth > 2 THEN                    0670
               levchg ← (curdpth - 1) -
               curblk/levblk;                                     0671
            IF levchg > curdpth THEN levchg ←
            curdpth;                                              0672
```

```
                        curdpth ← curdpth - levchg;                    0673
                        rlevcnt ← levsuc;                              0674
                        WHILE (levchg ← levchg - 1) >= 0 DO            0675
                           rlevcnt ← rlevcnt + 1;                      0676
                        END;                                           0677
                     END                                               0678
                  ELSE                                                 0679
                     BEGIN                                             0680
                     rlevcnt ← levsuc;                                 0681
                     IF lstblk THEN                                    0682
                        WHILE (curdpth := curdpth - 1) > 0 DO          0683
                           rlevcnt ← rlevcnt + 1;                      0684
                     curdpth ← 0;                                      0685
                     END;                                              0686
                  lstblk ← curblk;                                     0687
                  END;                                                 0688
               ENDCASE *lit* ← *lit*, char;                            0689
      IF NOT isread(jfn, $sar, CR) THEN EXIT LOOP;                     0690
      pos ← empty;                                                     0691
      END;                                                             0692
   IF lit.L NOT= empty THEN                                            0693
      BEGIN                                                            0694
      FIND $F(*lit*) ↑start $E(*lit*) ↑end;                            0695
      stid ← <STRMNP, cinssta> (stid, rlevcnt, $start, $end);          0696
      END;                                                             0697
   %close file%                                                        0698
      (inseqc1):                                                       0699
      IF NOT sysclose (jfn, $lit) THEN                                 0700
         BEGIN                                                         0701
         reljfn (jfn);                                                 0702
         err($lit);                                                    0703
         END;                                                          0704
   dismes(0);                                                          0705
   IF firststid=0 THEN firststid←stid;                                02284
   RETURN(firststid);                                                  0706
                                                                       0707
   (inseqaws):                                                         0708
                                                                       0709

   % the following code is concerned with insert sequential
   assembler file with structure.  the first statment in the
   assembler file is inserted without modification at the level
   specified by the user.  succeeding statements are placed one
   level levdown for each tab at the front of the statement and up
   the appropriate number of levels if a succeeding statement has
   less tabs than the current statement.  ASCIZ, ASCII, and SIXBIT
   statements are handled as follows:  the first non-blank
   character following the delimiter character for the ASCIZ, ASCII,
   and SIXBIT pseudo-ops serves as a delimiter.  all characters
   between this delimiter and its next occurrence are inserted
   literally in the statement, i.e., EOL or CR do not serve to
   delimit the statement, and no level adjustments are made. %      0710
                                                                    0711
   % insert the first statement %                                   0712
                                                                    0713
   *lit* ← NULL;                                                    0714
   IF NOT isread(jfn, $sar, CR) THEN GOTO iswsend;                  0715
   pos ← empty + 1;
```

```
   WHILE *sar*[pos] = LF  DO BUMP pos;                                  0716
   curdpth ← 0;                                                         0717
   LOOP                                                                 0718
      CASE char ← *sar*[pos := pos + 1]  OF                             0719
         = LF: NULL;                                                    0720
         = ENDCHR:  % the only way we get here is end of file %         0721
            BEGIN                                                       0722
            IF lit.L NOT= empty THEN                                    0723
               BEGIN                                                    0724
               FIND SF(*lit*) ↑start SE(*lit*) ↑end;                    0725
               cinssta(stid, rlevcnt, $start, $end);                    0726
               END;                                                     0727
            GOTO inseqcl;  % go close the file %                        0728
            END;                                                        0729
         = EOL, =CR:                                                    0730
            BEGIN                                                       0731
            FIND SF(*lit*) ↑start SE(*lit*) ↑end;                       0732
            stid ← cinssta(stid, rlevcnt, $start, $end);                0733
            EXIT LOOP;                                                  0734
            END;                                                        0735
         ENDCASE *lit* ← *lit*, char;                                   0736
                                                                        0737
   LOOP   % this is the main loop for the rest of the statements %      0738
      BEGIN                                                             0739
                                                                        0740
      % initialization %                                                0741
      *lit* ← NULL;                                                     0742
      rlevcnt ← levsuc;                                                 0743
                                                                        0744
      % read in a statement from the assembler file %                   0745
      IF NOT isread(jfn, $sar, CR) THEN EXIT LOOP;                      0746
         % i.e. GOTO iswsend %                                          0747
      pos ← empty + 1;                                                  0748
                                                                        0749
      % bypass LF %                                                     0750
      WHILE *sar*[pos] = LF DO BUMP pos;                                0751
                                                                        0752
      % find out level of statement relative to previous statement %    0753
      count ← 0;                                                        0754
      WHILE *sar*[pos] = TAB DO                                         0755
         BEGIN                                                          0756
         BUMP count;                                                    0757
         BUMP pos;                                                      0758
         END;                                                           0759
      CASE count OF                                                     0760
         > curdpth:                                                     0761
            BEGIN                                                       0762
            rlevcnt ← levdown;                                          0763
            count ← count - curdpth;                                    0764
            curdpth ← curdpth + 1;                                      0765
            WHILE (count ← count - 1) > 0 DO *lit* ← *lit*, TAB;        0766
            END;                                                        0767
         < curdpth:                                                     0768
            BEGIN                                                       0769
```

```
        t1 ← curdpth - count;                                                   0770
        WHILE (t1 := t1 - 1 ) > 0 DO rlevcnt ← rlevcnt +1;                      0771
        curdpth ← count;                                                        0772
        END;                                                                    0773
    ENDCASE;                                                                    0774
                                                                                0775
% we are now past any leading tabs and will gobble up the rest
of the statement one character at a time %                                      0776
                                                                                0777
LOOP                                                                            0778
    CASE char ← *sar*[pos := pos + 1] OF                                        0779
        = LF: NULL;
        = ENDCHR:  % the only way we get here is end of file %                  0780
                                                                                0781
            BEGIN                                                               0782
            IF lit.L NOT= empty THEN                                            0783
                BEGIN                                                           0784
                FIND SF(*lit*) ↑start SE(*lit*) ↑end;                           0785
                cinssta(stid, rlevcnt, $start, $end);                           0786
                END;                                                            0787
            GOTO inseqcl;  % go close the file %                                0788
            END;                                                                0789
        = EOL, = CR:                                                            0790
            BEGIN                                                               0791
            FIND SF(*lit*) ↑start SE(*lit*) ↑end;                               0792
            stid ← cinssta(stid, rlevcnt, $start, $end);                        0793
            EXIT LOOP;                                                          0794
            END;                                                                0795
        = 'A, ='a:   % check for asciz or ascii %                               0796
            BEGIN                                                               0797
            *lit* ← *lit*, char;                                                0798
            IF pos + 4 <= sar.L THEN                                            0799
                BEGIN                                                           0800
                t1 ← *sar*[pos + 0];                                            0801
                t2 ← *sar*[pos + 1];                                            0802
                t3 ← *sar*[pos + 2];                                            0803
                t4 ← *sar*[pos + 3];                                            0804
                t5 ← *sar*[pos + 4];                                            0805
                IF ( (t1 = 'S) OR (t1 = 's) ) AND                               0806
                   ( (t2 = 'C) OR (t2 = 'c) ) AND                               0807
                   ( (t3 = 'I) OR (t3 = 'i) ) AND                               0808
                   ( (t4 = 'Z) OR (t4 = 'z) OR                                  0809
                     (t4 = 'I) OR (t4 = 'i) ) AND                               0810
                   ( (t5 NOT IN ['A, 'Z]) AND                                   0811
                     (t5 NOT IN ['a, 'z]) AND                                   0812
                     (t5 # '$) AND (t5 # '%) AND                                0813
                     (t5 # '.) AND (t5 # LF) AND                                0814
                     (t5 # EOL) AND (t5 # CR) )  THEN                           0815
                   BEGIN                                                        0816
                   *lit* ← *lit*, t1, t2, t3, t4, t5;                           0817
                   pos ← pos + 4;                                               0818
                   IF (t5 # SP) AND (t5 # TAB) THEN term ← t5                   0819
                   ELSE LOOP                                                    0820
                      IF (pos ← pos + 1) <= sar.L THEN                          0821
                         IF ((term ← *sar*[pos]) = SP) OR
                            (term = TAB) THEN *lit* ← *lit*, term               0822
```

```
                        ELSE                                            0823
                           BEGIN                                        0824
                           *lit* ← *lit*, term;                         0825
                           EXIT LOOP;                                   0826
                           END                                          0827
                     ELSE                                               0828
                        BEGIN                                           0829
                        IF NOT isread(jfn, $sar, CR) THEN               0830
                           GOTO iswsend;                                0831
                        pos ← empty;                                    0832
                        END;                                            0833
                                                                        0834
                  % we have now gobbled up asciz or ascii and the
                  first delimiter and are gobbling succeeding
                  characters until we find the second occurrence
                  of the delimiter %                                    0835
                  LOOP                                                  0836
                     CASE char ← *sar*(pos ← pos + 1) OF                0837
                        = LF: NULL;                                     0838
                                                                        0839
                        % the only way we get here is end of
                        file%                                           0840
                        = ENDCHR:                                       0841
                           BEGIN                                        0842
                           IF lit.L NOT= empty THEN                     0843
                              BEGIN                                     0844
                              FIND SF(*lit*) ↑start SE(*lit*)
                              ↑end;                                     0845
                              cinssta(stid, rlevcnt, $start,
                              $end);                                    0846
                              END;                                      0847
                           GOTO inseqcl;  % go close the file %
                                                                        0848
                           END;                                         0849
                        = EOL, = CR:                                    0850
                           BEGIN                                        0851
                           *lit* ← *lit*, EOL;                          0852
                           IF NOT isread(jfn, $sar, CR) THEN            0853
                              GOTO iswsend;                             0854
                           pos ← empty;                                 0855
                           END;                                         0856
                        = term:                                         0857
                           BEGIN                                        0858
                           *lit* ← *lit*, char;                         0859
                           pos ← pos + 1;                               0860
                           EXIT LOOP;                                   0861
                           END;                                         0862
                        ENDCASE *lit* ← *lit*, char;                    0863
                  END;                                                  0864
               END;                                                     0865
            END;                                                        0866
      = 'S, ='s:   % check for sixbit %                                 0867
            BEGIN                                                       0868
            *lit* ← *lit*, char;                                        0869
            IF pos + 5 <= sar.L THEN                                    0870
               BEGIN                                                    0871
```

```
t1 ← *sar*[pos + 0];                                              0872
t2 ← *sar*[pos + 1];                                              0873
t3 ← *sar*[pos + 2];                                              0874
t4 ← *sar*[pos + 3];                                              0875
t5 ← *sar*[pos + 4];                                              0876
t6 ← *sar*[pos + 5];                                              0877
IF ( (t1 = 'I) OR (t1 = 'i) ) AND                                 0878
   ( (t2 = 'X) OR (t2 = 'x) ) AND                                 0879
   ( (t3 = 'B) OR (t3 = 'b) ) AND                                 0880
   ( (t4 = 'I) OR (t4 = 'i) ) AND                                 0881
   ( (t5 = 'T) OR (t4 = 't) ) AND                                 0882
   ( (t6 NOT IN ['A, 'Z]) AND                                     0883
     (t6 NOT IN ['a, 'z]) AND                                     0884
     (t6 # '$) AND (t6 # '%) AND                                  0885
     (t6 # '.) AND (t6 # LF) AND                                  0886
     (t6 # EOL) AND (t6 # CR) ) THEN                              0887
   BEGIN                                                          0888
   *lit* ← *lit*, t1, t2, t3, t4, t5, t6;                         0889
   pos ← pos + 5;                                                 0890
   IF (t6 # SP) AND (t6 # TAB) THEN term ← t6                     0891
   ELSE LOOP                                                      0892
      IF (pos ← pos + 1) <= sar.L THEN                            0893
         IF ((term ← *sar*[pos]) = SP) OR                         0894
            (term = TAB) THEN *lit* ← *lit*, term
                                                                  0895
         ELSE                                                     0896
            BEGIN                                                 0897
            *lit* ← *lit*, term;                                  0898
            EXIT LOOP;                                            0899
            END                                                   0900
      ELSE                                                        0901
         BEGIN                                                    0902
         IF NOT isread(jfn, $sar, CR) THEN                        0903
            GOTO iswsend;                                         0904
         pos ← empty;                                             0905
         END;                                                     0906
                                                                  0907
   % we have now gobbled up sixbit and the first
   delimiter and are gobbling succeeding
   characters until we find the second occurrence
   of the delimiter %                                             0908
   LOOP                                                           0909
      CASE char ← *sar*[pos ← pos + 1] OF                         0910
         = LF: NULL;                                              0911
                                                                  0912
         % the only way we get here is end of
         file%                                                    0913
         = ENDCHR:                                                0914
            BEGIN                                                 0915
            IF lit.L NOT= empty THEN                              0916
               BEGIN                                              0917
               FIND SF(*lit*) ↑start SE(*lit*)
               ↑end;                                              0918
               cinssta(stid, rlevcnt, $start,
               $end);                                             0919
               END;                                               0920
```

```
                                GOTO inseqc1;  % go close the file %
                                END;
                             = EOL, = CR:
                                BEGIN
                                *lit* ← *lit*, EOL;
                                IF NOT isread(jfn, $sar, CR) THEN
                                   GOTO iswsend;
                                pos ← empty;
                                END;
                             = term:
                                BEGIN
                                *lit* ← *lit*, char;
                                pos ← pos + 1;
                                EXIT LOOP;
                                END;
                             ENDCASE *lit* ← *lit*, char;
                          END;
                       END;
                    END;
                 ENDCASE *lit* ← *lit*, char;
           END;

      (iswsend):   % cleanup at end of file %
      tl ← sar.L;
      pos ← empty;
      WHILE (tl := tl - 1) > 0 DO *lit* ← *lit*, *sar*[pos ← pos + 1];

      IF lit.L NOT= empty THEN
         BEGIN
         FIND SF(*lit*) ↑start SE(*lit*) ↑end;
         cinssta(stid, rlevcnt, $start, $end);
         END;
      GOTO inseqc1;   % close the file and return %

      END.

(isread)
      %This procedure is used during insert sequential to read another
      character from the input file.  It expects the input jfn, the
      address of a string, and a terminating character, as arguments,
      in that order.  It puts the characters read into the string; it
      returns true until it has an empty input buffer and encounters an
      EOF.%
      %--------------------%
      PROCEDURE(jfn, inbuff, lstchr);
      LOCAL error, maxcnt, char, count, bytptr;
      REF inbuff;
      bytptr ← chbmty + &inbuff;
      maxcnt ← inbuff.M;
      count ← empty;
      LOOP
         BEGIN
         !bin( jfn );
         IF NOT (char ← r2 .A 377B) % null byte % THEN
            BEGIN
```

```
        lgtsts( jfn );                                                  0970
        error ← r2;                                                     0971
        IF error .A seqer # 0 THEN% Input error %                       0972
           BEGIN                                                        0973
           IF NOT sysclose (jfn, $lit) THEN                             0974
              BEGIN                                                     0975
              reljfn (jfn);                                             0976
              err ($lit);                                               0977
              END                                                       0978
           ELSE err ($"Input File Error");                              0979
           END;                                                         0980
        IF error .A seqeof # 0 THEN % EOF read-- check if buffer
        empty %                                                         0981
           BEGIN                                                        0982
           IF (inbuff.L ← count) > empty THEN RETURN(TRUE)              0983
           ELSE RETURN( FALSE );                                        0984
           END;                                                         0985
        END;                                                            0986
     ↑bytptr ← char;                                                    0987
     IF ((count ← count + 1) >= maxcnt) OR (char = lstchr) OR (char
     = EOL AND lstchr = CR) THEN                                        0988
        BEGIN                                                           0989
        inbuff.L ← count;                                               0990
        RETURN(TRUE);                                                   0991
        END;                                                            0992
     END;                                                               0993
  END.
                                                                        0994
(inseqn) PROCEDURE   %does the input sequential command (new way):
double CR means end of st.%                                             0995
   (stid,       % statement after which to insert                       %
                                                                        0996
    rlevcnt,    % level relative to stid                                 %
                                                                        0997
    ifilnam,    % string containing name of input file                   %
                                                                        0998
    just);      % were multiple spaces added to right justify?          0999
   ------------------------------------------------------------------%
                                                                        01000
   % declarations %                                                    01001
      LOCAL jfn, place, dummy, indent1, indent2, level, adj, done ;
                                                                        01002
      LOCAL TEXT POINTER stop, end, t1, t2, z1, z2 ;                    01003
      LOCAL STRING line1[999], line2[999], st[2000], string[20] ;
                                                                        01004
      REF ifilnam ;                                                     01005
      level ← done ← 0;                                                 01006
   dismes (1, $"Insert Sequential in Progress ");                       01007
   % handle different types %                                          01008
   % open input file %                                                 01009
      IF NOT jfn ← lgetjfn (0, &ifilnam, $txtext, gtjoisf, $lit)
                                                                        01010
         THEN SIGNAL (ofilerr, @lit) ;                                  01011
      IF NOT sysopen (jfn, read, chrtyp, $lit) THEN                     01012
         BEGIN                                                          01013
         reljfn (jfn) ;                                                 01014
```

```
        SIGNAL (ofilerr, $lit) ;                                        01015
        END;                                                            01016
    % insert dummy statement %                                          01017
      *string* ← "dummy";                                               01018
      FIND SF(*string*) ↑t1 SE(*string*) ↑t2;                           01019
      dummy ← place ← cinssta (stid, rlevcnt, $t1, $t2) ;               01020
    % for all data in input file %                                      01021
      LOOP %each cycle a statement%                                     01022
        BEGIN                                                           01023
        adj ← indent1 ← indent2 ← 0 ;                                   01024
        % get first line %                                              01025
          DO %skip blank lines%                                         01026
            BEGIN                                                       01027
            indent1 ← inseq1 (jfn, $line1, just) ;                      01028
            IF indent1 = -1 THEN                                        01029
              BEGIN                                                     01030
              done ← 1;                                                 01031
              EXIT LOOP;                                                01032
              END;                                                      01033
            END                                                         01034
          UNTIL line1.L>0 ;                                             01035
        % get second line %                                             01036
          indent2 ← inseq1 (jfn, $line2, just) ;                        01037
          IF indent2 = -1 THEN                                          01038
            done ← 1;                                                   01039
        % add both lines to *st* %                                      01040
          *st* ← *line1*, *line2* ;                                     01041
        % figure out level %                                            01042
          IF line2.L > 0 THEN indent1 ← MIN (indent1, indent2);         01043
          CASE indent1 OF                                               01044
            > levind[level]:                                            01045
              IF level = 11 THEN                                        01046
                adj ← levsuc                                            01047
              ELSE                                                      01048
                BEGIN                                                   01049
                BUMP level;                                             01050
                levind[level] ← indent1 ;                               01051
                IF (place := getsub(place)) = place                     01052
                  THEN adj ← levdown                                    01053
                  ELSE                                                  01054
                    BEGIN                                               01055
                    place ← getail(place);                              01056
                    adj ← levsuc;                                       01057
                    END;                                                01058
                END;                                                    01059
            < levind[level]:                                            01060
              BEGIN                                                     01061
              BUMP DOWN level ;                                         01062
              place ← getup (place) ;                                   01063
              REPEAT CASE;                                              01064
              END;                                                      01065
            ENDCASE  % =levind[level] %                                 01066
              adj ← levsuc ;                                            01067
        % add lines to *st* until double EOL %                          01068
          IF line2.L > 0 THEN                                           01069
```

```
                LOOP                                                    01070
                   BEGIN                                                01071
                   IF inseql(jfn,$line1,just) = -1 THEN                 01072
                      BEGIN                                             01073
                      done ← 1;                                         01074
                      EXIT LOOP;                                        01075
                      END;                                              01076
                   IF line1.L=0 THEN EXIT LOOP;                         01077
                   IF (st.L + line1.L) > st.M THEN                      01078
                      BEGIN  %start new statement%                      01079
                      FIND SF(*st*) ↑z1 SE(*st*) ↑z2;                   01080
                      place ← cinssta (place, adj, $z1, $z2) ;          01081
                      adj ← levsuc ;                                    01082
                      st.L ← 0 ;                                        01083
                      END;                                              01084
                   *st* ← *st*, *line1* ;                               01085
                   END;                                                 01086
             % insert the statement %                                   01087
                FIND SE(*st*) ↑end $NP ↑stop > ;                        01088
                ST stop end ← NULL ;                                    01089
                FIND SF(*st*) ↑z1 SE(*st*) ↑z2;                         01090
                place ← cinssta (place, adj, $z1, $z2) ;                01091
                IF done THEN EXIT LOOP;                                 01092
             END;                                                       01093
      % close the input file %                                          01094
         IF NOT sysclose (jfn, $lit) THEN                               01095
             BEGIN                                                      01096
             reljfn (jfn) ;                                             01097
             dismes (2, $lit) ;                                         01098
             GOTO STATE;                                                01099
             END;                                                       01100
      % delete dummy statement %                                        01101
         IF (place ← getsub(dummy)) # dummy THEN                        01102
             cmovgro (dummy, levsuc, gethed(place) , getail(place),
             FALSE, 0) ;                                                01103
         cdelsta (dummy, FALSE, 0) ;                                    01104
      dismes(0) ;                                                       01105
      RETURN;                                                           01106
      END.                                                              01107
(inseql) PROCEDURE (jfn, line, justified) ; %gets a line for inseqn%
                                                                        01108
      % declarations %                                                  01109
         LOCAL indent, short ;                                          01110
         LOCAL TEXT POINTER pt1, pt2 ;                                  01111
         REF line ;                                                     01112
      % get line %                                                      01113
         line.L ← 0 ;                                                   01114
         IF NOT lsread (jfn, &line, CR) THEN RETURN (-1); %couldn't get
         line%                                                          01115
         short ← (IF line.L < 63 THEN TRUE ELSE FALSE) ;                01116
      % if within 8 chars of RM, and if fully justified, remove extra
      spaces if less than or equal to 5 in a row %                      01117
         IF justified AND NOT short THEN                                01118
             BEGIN                                                      01119
             FIND SF(*line*) $NP ;                                      01120
             LOOP                                                       01121
```

```
            BEGIN                                                   01122
            IF NOT FIND [2SP] THEN EXIT LOOP;                       01123
            IF FIND ↑pt1 ←pt1 $3SP ↑pt2 PT THEN                     01124
               ST pt1 pt2 ← NULL                                    01125
            ELSE FIND $SP;                                          01126
            END;                                                    01127
         END;                                                       01128
   % count and delete leading spaces %                              01129
      CCPOS SF(*line*) ;                                            01130
      indent ← 0;                                                   01131
      CASE READC OF                                                 01132
         =SP:                                                       01133
            BEGIN                                                   01134
            indent ← indent + 1 ;                                   01135
            REPEAT CASE ;                                           01136
            END;                                                    01137
         =TAB:                                                      01138
            BEGIN                                                   01139
            indent ← indent + 8 ;                                   01140
            REPEAT CASE ;                                           01141
            END;                                                    01142
         =NP:  REPEAT CASE;                                         01143
         ENDCASE FIND ↑pt1 ←pt1;                                    01144
   % delete trailing spaces %                                       01145
      CCPOS SE(*line*) ;                                            01146
      CASE READC OF                                                 01147
         =CR:                                                       01148
            IF short THEN                                           01149
               BEGIN                                                01150
               FIND ↑pt2 ←pt2 >;                                    01151
               *line* ← pt1 pt2 ;                                   01152
               IF line.L=1 THEN line.L ← 0 ;                        01153
               END                                                  01154
            ELSE REPEAT CASE ;                                      01155
         =NP:  REPEAT CASE;                                         01156
         ENDCASE                                                    01157
            BEGIN                                                   01158
            FIND ↑pt2 ←pt2 >;                                       01159
            *line* ← pt1 pt2, SP ;                                  01160
            END;                                                    01161
   RETURN (indent);                                                 01162
   END.                                                             01163
(hiop) PROCEDURE (acs); % NLS Host Input/Output Processor %         01164
   % Declarations %                                                 01165
      LOCAL                                                         01166
         direction, % file entering/leaving system %                01167
         parmchr,   % single char of parms %                        01168
         i,         % working index variable %                      01169
         injfn,     % source file jfn %                             01170
         outjfn,    % result file jfn %                             01171
         bp1, bp2;  % byte pointers %                               01172
      LOCAL TEXT POINTER                                            01173
         tp1, tp2, tp3, tp4, z1, z2; % working pointers %           01174
      LOCAL STRING                                                  01175
         parms [40],      % parameters %                           01176
         tail [40],       % tail of parms %                        01177
```

```
        infile [40],     % source filename %                          01178
        outfile [40],    % result filename %                          01179
        fieldname [40], % name of field %                            01180
        fieldvalue [40],% value of field %                           01181
        viewspecs [40], % viewspecs string %                         01182
        addr [40],       % statement addr %                          01183
        convtype [40],   % conversion algorithm %                    01184
        errstr [40];     % error string %                            01185
    REF acs; % AC's upon entry to NLS %                              01186
% Save direction and JFN %                                          01187
    ON SIGNAL ELSE                                                   01188
        hiortn (cat, $"Conversion setup error: ", $sysmsg, $""); %
        catch signals %                                              01189
    r1 ← acs [7]; % get xwd direction, input jfn %                   01190
    ! HLRE 2,1; % isolate direction, propagate sign %                01191
    ! HRRZS 1; % clean input jfn %                                   01192
    injfn ← r1; % save input jfn %                                   01193
    direction ← r2; %save direction %                                01194
% Build parameter string %                                          01195
    bp1 ← chbptr(empty) + $parms;                                    01196
    bp2 ← chbptr(empty) + &acs + 7;                                  01197
    r1 ← bp1; % load destination AC %                                01198
    r2 ← bp2; % load source AC %                                     01199
    r3 ← 0; % terminate on NUL %                                     01200
    ! JSYS sout; % copy the ASCIZ string %                          01201
    bp2 ← r1; % save dest-end pointer %                              01202
    parms.L ← slngth (bp1, bp2); % set parms length %               01203
% Init defaults %                                                   01204
    *viewspecs* ← ""; % null viewspecs %                            01205
    *convtype* ← "S"; % convert to/from sequential file %           01206
    *addr* ← ".1"; % point to top of file %                         01207
% Extract fields from parameter string %                            01208
    CCPOS *parms*; % init text pointer %                             01209
    *tail* ← *parms*; % init tail of parms %                        01210
    IF parms.L # 0 THEN % skip parse if null parms %                 01211
        LOOP                                                         01212
            BEGIN % here for each field of parms %                   01213
            IF NOT FIND ↑tp1 $LD ↑tp2 ': ↑tp3 ((/',/] ↑tp4 ←tp4) /
            (SE(tp3) ↑tp4 >)) THEN hiortn (cat, $"What is '", $tail,
            $"'?"); % extract next field and its value %             01214
            *fieldname* ← + tp1 tp2; % save name of field %          01215
            *fieldvalue* ← tp3 tp4; % and its value %                01216
            IF fieldname.L # 1 THEN GOTO badfield; % all field types
            are one char %                                           01217
            CASE *fieldname* [1] OF % save field value appropriately
            %                                                        01218
                ='T: *convtype* ← + SF(*fieldvalue*)
                SE(*fieldvalue*); % conversion type %                01219
                ='V: *viewspecs* ← *fieldvalue*; % viewspecs %       01220
                ='N: *addr* ← *fieldvalue*; % statement %            01221
                ENDCASE (badfield): hiortn (cat, $fieldname, $"-field
                undefined", $""); % undefined field type %          01222
            *tail* ← tp4 SE(tp4); % parms all processed? %           01223
            IF tail.L = 0 THEN EXIT LOOP; % quit if so %             01224
            END;                                                     01225
% Load source file %                                                01226
```

```
        ON SIGNAL ELSE                                                  01227
           hiortn (cat, $"Can't load file: ", $sysmsg, $""); % catch
           signals %                                                    01228
        jfnstr (injfn, $infile); % fetch filename %                     01229
        tda.dacsp ← orgstid;  tda.daccnt ← 1;                           01230
        tda.dacsp.stfile ← cloafil ($infile); % load source file %
                                                                        01231
     % Prepare output file %                                           01232
        ON SIGNAL ELSE                                                  01233
           hiortn (cat, $"Can't get output file: ", $sysmsg, $""); %
           catch signals %                                              01234
        IF NOT outjfn ← sgtjfn (6B11, $"HIOPWORK.TXT", $errstr) THEN
        hiortn (cat, $"Problems with output file: ", $errstr, $""); %
        abort if problems with output file %                            01235
        jfnstr (outjfn, $outfile); % fetch output filename %            01236
     % Verify and instate viewspecs %                                  01237
        ON SIGNAL ELSE                                                  01238
           hiortn (cat, $viewspecs, $" are invalid viewspecs: ",
           $sysmsg); % catch signals %                                  01239
        feedlt (&tda, $viewspecs); % build viewspecs %                  01240
     % Verify and position to statement addr %                         01241
        ON SIGNAL ELSE                                                  01242
           hiortn (cat, $addr, $" is an invalid addr: ", $sysmsg); %
           catch signals %                                              01243
        bl ← tda.dacsp; % get current... %                              01244
        bl [1] ← tda.daccnt; % ...command marker %                      01245
        FIND SF(*addr*) ↑z1 SE(*addr*) ↑z2;                             01246
        caddexp ($z1, $z2, &tda, $bl); % interpret TNLS addr %          01247
        tda.dacsp ← bl;  tda.daccnt ← bl[1];                            01248
     % Invoke the conversion %                                         01249
        ON SIGNAL ELSE                                                  01250
           hiortn (cat, $convtype, $"-conversion error: ", $sysmsg); %
           catch signals %                                              01251
        IF direction # leaving THEN hiortn (cat, $"Input not yet
        supported", $"", $""); % file input not yet implemented %       01252
        CASE *convtype* [1] OF                                          01253
           ='A: outasm ($outfile, &tda, FALSE);                         01254
           ='P: outproc ($outfile, &tda, opprdv, 0);                    01255
           ='Q: outqp ($outfile, &tda);                                 01256
           ='S: outseq ($outfile, &tda, FALSE);                         01257
           ENDCASE hiortn (cat, $convtype, $"-conversion type
           undefined", $"");                                            01258
        hiortn (outjfn, $convtype, $"-conversion", $""); % successful
        completion %                                                    01259
        END.                                                            01260
%.....output device and output compiler.....%
                                                                        01261
%.....processor dispatch.....%
                                                                        01262
    (processor) PROCEDURE   % start and run a processor %               01939
       (prcnam,  % string containing the name of the processor %        01940
       tp,       % text pointer to first thing to feed the processor %
                                                                        01941
       da,       % display area descriptor of the input to processor %
                                                                        01942
       type,     % type of the processor %                             01943
```

```
outdsg,   % output designator:  either the address of a buffer or
the jfn of an output file %                                          01944
prcjfn);  %JFN of processor if no processor name pased or 0% 01945
                                                                     01946
   % ****** The Catalog System has its own version of PROCESSOR
   so please tell Walt if you make any changes ****** %              01947
                                                                     01948
                                                                     01949
LOCAL stid, size, error, savpreg, savwareg, vspec, sdb;              01950
LOCAL TEXT POINTER fchartp, tptr;                                    01951
LOCAL STRING ssysnam[40];                                            01952
REF prcnam, tp, da, sdb;                                             01953
                                                                     01954
% set up to get first character for the processor %                  01955
   vspec ← da.davspec;                                               01956
   IF type = cmptyp THEN vspec.vsstnf ← TRUE;                        01957
      % turn statement numbers on for MPL %                         01958
   &prseqwk  ← openseq (tp, seqend(tp, vspec, da.davspc2), vspec,
   da.davspc2, da.dausqcod, da.dacacode);                            01959
   IF (stid ← seqgen(&prseqwk)) = endfil THEN                        01960
      BEGIN                                                          01961
      closeseq (&prseqwk);                                           01962
      SIGNAL (prcerr, $"No processor input");                        01963
      END;                                                           01964
   IF tp.stastr THEN FIND (tp ↑fchartp)                              01965
   ELSE FIND (SF(stid) ↑fchartp);                                    01966
IF NOT prcjfn AND NOT (prcjfn ← lgetjfn($subdir, &prcnam,
$savext, gtjpri, $lit))                                              01967
   THEN SIGNAL (prcerr, $lit);                                       01968
% get a string containing name of the processor %                    01969
   IF type = upgtyp THEN *ssysnam* ← "NLSLIO"                        01970
   ELSE                                                              01971
      BEGIN                                                          01972
      jfntostr (prcjfn, $ssysnam, 11B9);                             01973
      IF NOT FIND SF(*ssysnam*) "SUBSYS" ↑tptr THEN                  01974
         *ssysnam* ← "PRVPRC"                                        01975
      ELSE *ssysnam* ← tptr SE(*ssysnam*);                           01976
      END;                                                           01977
%map out high pages%                                                 01978
   hmapout();                                                        01979
% set the subsystem name %                                          01980
   r1 ← getsbn ($ssysnam);                                           01981
   !JSYS setnm;                                                      01982
% "get" the processor %                                             01983
   r1.LH ← 4B5;                                                      01984
   r1.RH ← prcjfn;                                                   01985
   !JSYS get;                                                        01986
IF [prcadr + 1] THEN chntab[9] ← [prcadr + 1] .V 1B6;                01987
savpreg ← p; savwareg ← wa;  % save wa and p registers since the
compilers and the Output Processor will clobber them %               01988
ON SIGNAL ELSE GOTO procrestcre;                                     01989
CASE type OF                                                         01990
   = upgtyp:   % compile a user program %                            01991
      BEGIN                                                          01992
      IF NOT cppflag THEN dismes (1, $"Compiling User Program");
                                                                     01993
```

```
        swork ← fchartp; swork[1] ← fchartp[1]; fechcl(1, $swork);
                                                                    01994
        %swork[2] contains byte count+1, swork[4] has byte pointer%
                                                                    01995
        error ← [[prcadr]]                                         01996
           (-1, $prgetst, outdsg, , $levllc, swork[4], swork[2]-1
           :size);                                                  01997
        END;                                                        01998
     = cmptyp:   % compile a program into an output file %          01999
        BEGIN                                                       02000
        dismes (1, $"Compiling");                                   02001
        swork ← fchartp; swork[1] ← fchartp[1]; fechcl(1, $swork);
                                                                    02002
        %swork[2] contains byte count+1, swork[4] has byte pointer%
                                                                    02003
        error ← [[prcadr]]                                         02004
           (outdsg, $prgetst, prseqwk.swsvw, , $levllc, swork[4],
           swork[2]-1);                                             02005
        END;                                                        02006
     = odtyp:    % output device type %                             02007
        BEGIN                                                       02008
        dismes (1, IF exp                                           02009
           THEN $"Experimental Output Processor"                    02010
           ELSE $"Processing Output");                              02011
        lodsdb(getsdb(stid) :&sdb);                                 02012
        opbp ← opcbp ← &sdb +sdbhdl + 4407B8;                       02013
        opcmax ← sdb.schars + 1;                                    02014
        opccpos ← 1;                                                02015
        opnewst ← TRUE;                                             02016
        error ← [[prcadr]] ($oprwrk, $levllc, opbp, opcmax);       02017
        END;                                                        02018
     ENDCASE   % some other type of processor %                     02019
        BEGIN                                                       02020
        dismes (1, $"Processor in progress");                       02021
        swork ← fchartp; swork[1] ← fchartp[1]; fechcl(1, $swork);
                                                                    02022
        %swork[2] contains byte count+1, swork[4] has byte pointer%
                                                                    02023
        error ← [[prcadr]] (outdsg, $prgetst, , , $levllc,
        swork[4], swork[2]-1);                                      02024
        END;                                                        02025
(procrestore):                                                      02026
% restore wa and p registers %                                     02027
   p ← savpreg; wa ← savwareg;                                      02028
%restore stack overflow pseudo-interrupt%                           02029
   chntab[9] ← $stkovr .V 1B6;                                      02030
%get nls pages back%                                                02031
   hmapin();                                                        02032
ON SIGNAL ELSE                                                      02033
   BEGIN                                                            02034
   ON SIGNAL ELSE; % In case closeseq generates a signal %          02035
   %close the sequence%                                             02036
      closeseq (&prseqwk);                                          02037
   inptrf ← 0;                                                      02038
   END;                                                             02039
%restore subsystem name to NLS%                                     02040
```

```
      r1 ← nlssbn;                                                    02041
      ↓JSYS setnm;                                                    02042
   %take message away%                                                02043
      dismes (0);                                                     02044
   %close the sequence%                                               02045
      closeseq (&prseqwk);                                            02046
   %reset entry vector back to NLS%                                   02047
      setvec();                                                       02048
   IF inptrf THEN %to pseudo-interrupt%                               02049
      BEGIN                                                           02050
      inptrf ← 0;                                                     02051
      clrbuf (1);                                                     02052
      IF type # upgtyp THEN                                           02053
         IF NOT <ICEXEC, sysclose> (outdsg, $lit) THEN                02054
            dismes (2, $lit);                                         02055
      SIGNAL(-4, $"User Terminated Process");                         02056
      END;                                                            02057
   RETURN (error, size);                                              02058
   END.                                                               02059

(mapped) RECORD % pmap handle and page status. %                      01587
   fhand[26], % handle from handle. %                                 01588
   hndacc[10]; % access information. %                                01589
(hmapout)PROC;                                                        01590
   LOCAL count, fbase, handle;                                        01591
   %Keep track of pages mapped out%                                   01592
      FOR count ← (fbase ← $bfree/1000B) UP UNTIL >= $efree/1000B DO  01593
         BEGIN                                                        01594
         r1.LH ← 4B5;                                                 01595
         r1.RH ← count;                                               01596
         ↓JSYS rpacs;                                                 01597
         IF SKIP ↓TLNN r2, 10000B THEN %page exists%                  01598
            BEGIN                                                     01599
            %get handle for page%                                     01600
               ↓JSYS rmap;                                            01601
               handle ← r1;                                           01602
               handle.hndacc ← r2.hndacc;                             01603
            %remove page%                                             01604
               r2.LH ← 4B5;                                           01605
               r2.RH ← count;                                         01606
               r1 ← -1;                                               01607
               r3 ← 0;                                                01608
               ↓JSYS pmap;                                            01609
            END                                                       01610
         ELSE  %page does not exist%                                  01611
            handle ← -1;                                              01612
         freemap[count-fbase] ← handle;                               01613
         END;                                                         01614
   RETURN;                                                            01615
   END.                                                               01616

(hmapin)PROC;                                                         01638
   LOCAL count, fbase, hansav;                                        01639
   %get nls pages back%                                               01640
      FOR count ← (fbase ← $bfree/1000B) UP UNTIL >= $efree/1000B DO
```

```
                                                                    01441
        BEGIN                                                       01442
        IF ( hansav ← freemap[count-fbase]) # -1 THEN               01443
           BEGIN                                                    01444
           r1 ← hansav.fhand;                                       01445
           r3  ← 0;                                                 01446
           r3.hndacc ← hansav.hndacc;                               01447
           END                                                      01448
        ELSE                                                        01449
           BEGIN                                                    01450
           r1 ← -1;                                                 01451
           r3 ← 120400B6;                                           01452
           END;                                                     01453
        r2.LH ← 4B5;                                                01454
        r2.RH ← count;                                              01455
        !JSYS pmap;                                                 01456
        END;                                                        01457
   RETURN;                                                          01458
   END.                                                             01459
(prgetst) PROCEDURE;                                                02060
   % returns byte pointer + count for next statement %              02061
   LOCAL stid, sdb;                                                 02062
   REF sdb;                                                         02063
   IF (stid ← seqgen(&prseqwk)) = endfil THEN RETURN(endfil);       02064
   IF stid.stastr THEN RETURN(stid.stpsid+1+4407B8, [stid.stpsid].L)
                                                                    02065
   ELSE                                                             02066
      BEGIN                                                         02067
      lodsdb(getsdb(stid) : &sdb);
      return[&sdb + sdbhdl + 4407B8, sdb.schars);                   02069
      END;                                                          02070
   END.                                                             02071
                                                                    02072
(oprchr) PROCEDURE;          % get a character for OP %            01423
   IF opccpos = opcmax THEN RETURN(ENDCHR);                         01424
   BUMP opccpos;                                                    01425
   RETURN(↑opcbp);                                                  01426
   END.                                                             01427
                                                                    01428
(oprnst) PROCEDURE;          % get next statement for OP %          01429
   LOCAL stid, sdb;                                                 01430
   REF sdb;                                                         01431
   IF (stid ← seqgen(&prseqwk)) # endfil THEN                       01432
      BEGIN                                                         01433
      lodsdb(getsdb(stid) : &sdb);                                  01434
      opbp ← opcbp ← &sdb + sdbhdl + 4407B8;                        01435
      opcmax ← sdb.schars + 1;                                      01436
      opccpos ← 1;                                                  01437
      opnewst ← TRUE;                                               01438
      END;                                                          01439
   RETURN (stid, opbp, opcmax);                                     01440
   END.                                                             01441
                                                                    01442
(oprtxt) PROCEDURE;          % get char pos of 1st text after "name"
%                                                                   01443
   LOCAL TEXT POINTER tp;                                           01444
```

```
   IF prseqwk.swcstid.stastr THEN RETURN(FALSE);                    01445
   tp ← prseqwk.swcstid;                                            01446
   tp[1] ← fcntxt(getsdb(tp));                                      01447
   FOR tp[1] DOWN UNTIL <=0 DO                                      01448
      BEGIN                                                         01449
      !IBP opcbp;                                                   01450
      BUMP opccpos;                                                 01451
      END;                                                          01452
   RETURN(opccpos);                                                 01453
   END.                                                             01454
                                                                    01455
(oprgps) PROCEDURE;                                                 01456
   RETURN (opccpos);                                                01457
   END.                                                             01458
                                                                    01459
(oprrst) PROCEDURE  % set to passed char position %                 01460
   (pos);                                                           01461
   IF pos = -1 THEN opccpos ← opcmax                                01462
   ELSE opccpos ← pos;                                              01463
   opcbp ← chbptr(pos - 1) + opbp.RH -1;                            01466
   RETURN;                                                          01468
   END.                                                             01469
                                                                    01470
(oprlev) PROCEDURE;                                                 01471
   oplev ← prseqwk.swclvl;                                          01474
   RETURN ( oplev);                                                 01477
   END.                                                             01478
                                                                    01479
(oprsvc) PROCEDURE;                                                 01480
   stvect (prseqwk.swcstid, prseqwk.swsvw);                         01481
   RETURN (prseqwk.swsvw);                                          01482
   END.                                                             01483
                                                                    01484
(oprsig) PROCEDURE;                                                 01485
   *lit* ← NULL;                                                    01486
   fechsig (prseqwk.swcstid, $lit);                                 01487
   RETURN ($lit);                                                   01488
   END.                                                             01489
                                                                    01490
(oprhdf) PROCEDURE;                                                 01491
   RETURN (getfhd (prseqwk.swcstid));                               01492
   END.                                                             01493
                                                                    01494
(opinit) PROCEDURE (da, jfn, devtyp, opflags);                      01495
   oprwrk ← jfn;                                                    01496
   oprwrk[1] ← devtyp;                                              01497
   oprwrk[2] ← da;                                                  01498
   oprwrk[3] ← $oprchr;                                             01499
   oprwrk[4] ← $oprtxt;                                             01500
   oprwrk[5] ← $oprrst;                                             01501
   oprwrk[6] ← $oprgps;                                             01502
   oprwrk[7] ← $oprlev;                                             01503
   oprwrk[8] ← $oprsvc;                                             01504
   oprwrk[9] ← $oprsig;                                             01505
   oprwrk[10] ← $oprhdf;                                            01506
   oprwrk[11] ← $oprnst;                                            01507
```

```
    oprwrk[12] ← opflags;                                                01508
    RETURN;                                                              01509
    END.                                                                 01510
                                                                         01511
(getsbn) PROCEDURE (astrng);                                             01512
    %convert string passed to sixbit name%                               01513
    LOCAL sixbit, charct, bytptr;                                        01514
    REF astrng;                                                          01515
    bytptr ← 688 + $sixbit - 1;                                          01516
    sixbit ← 0;                                                          01517
    charct ← empty;                                                      01518
    WHILE ((charct ← charct + 1) <= 6) AND (charct <= astrng.L) DO
                                                                         01519
      ↑bytptr ← (*astrng*[charct] + 40B) .A 77B;                         01520
    RETURN(sixbit);                                                      01521
    END.
                                                                         01522
FINISH of SEQFIL                                                         01523
```