CIDOT

CIDENT CIDENT CIDENT CIDENT CIDENT CIDENT CIDENT CIDENT

```
CHI,   2-OCT-74 07:01          < NLS, CIDENT.NLS;14, >, 10-SEP-74 09:38 CHI ;;;;( NLS, CIDENT.NLS;12,
), 23-MAY-74 13:54 HGL ;
FILE cident % L10 to <REL-NLS>CIDENT %% (L10,) (rel-nls,CIDENT.rel,) %
                                                                                02
                                                                                03
%....declarations....%                                                          04
   DECLARE EXTERNAL
      allvers = 31, usedids = 1, inds = 2, groups = 4, orgs = 8, filver
                                                                                05
      = 16;                                                                    0129
%....generate tentative idents....%
   (gnxtid) %given an ident, generate a  new one by adding a number%
                                                                               0130
   PROCEDURE(idstr, fileno);                                                   0131
      LOCAL STRING numstr[5];                                                  0132
      LOCAL chrcnt;                                                            0133
      REF idstr;                                                               0134
      chrcnt ← idstr.L;                                                        0135
      *numstr* ← '2;                                                           0136
      LOOP                                                                     0137
         BEGIN                                                                 0138
         *idstr* ← *idstr*[1 TO chrcnt], *numstr*;                             0139
         IF NOT oldid(&idstr, fileno) THEN EXIT;                               0140
         bumpstr($numstr);                                                     0141
         END;                                                                  0142
      RETURN;
      END.                                                                     0143

   (getids) %expand list of idents into list of individuals%
   PROCEDURE (ptr, astr, infotype, idfile);                                    0144
      LOCAL [illegible], idf, ide, tmpptr, srcptr, dstptr;                     0146
      LOCAL STRING idstr[20], infostr[2000];                                   0147
      REF ptr, astr;                                                           0148
      %reads an ident from pointer ptr into astring idstr, and then
      calls ckident to get info on it.                                         0149
         IF infotype = 0, then it returns all of the info in astr, if
         infotype = 1, then the name only is returned.                         0150
         Uses infostr as a work area%                                          0151
                                                                               0152
      expchr ← 0;                                                              0153
      %first, read ident%                                                      0154
         LOOP                                                                  0155
            BEGIN                                                              0156
            CCPOS ptr;                                                         0157
            IF FIND $NP ↑idf '; THEN                                           0158
               BEGIN
               IF NOT popids(&ptr) THEN RETURN(FALSE); %no more idents%
                                                                               0159
               END                                                             0160
            ELSE EXIT LOOP;                                                    0161
            END;                                                               0162
         IF NOT FIND idf [ NP / '; / '( ]                                      0163
            ↑ptr ←ptr ↑ide ←ide THEN                                           0164
               err($"Ident List Format Error");                                0165
         FIND ptr ($NP '( [')] ↑ptr);                                          0166
         IF FIND idf ('&/'↑) ↑idf  < CH > THEN expchr ← READC;                 0167
         *idstr* ← idf ide; %ident%                                            0168
      %Now get info, and check ident%                                          0169
         IF ckident($idstr, $infostr, idfile : gpstid) THEN %return
```

```
        something%                                                          0170
           BEGIN                                                            0171
           IF orgrptst($infostr, 0) THEN                                    0172
              BEGIN                                                         0173
              expchr ← CASE expchr OF %expand group list or not%            0174
                 ='&: FALSE;                                                0175
                 ='↑: TRUE;                                                 0176
                 ENDCASE expdtst($infostr, 0); %take default from ident
                 record%                                                    0177
              IF expchr THEN                                                0178
                 BEGIN                                                      0179
                 getimem($infostr, 0,  $dstptr, 0);                         0180
                 IF FIND dstptr -EOL %membership list present% THEN
                                                                            0181
                    BEGIN                                                   0182
                    pushids(&ptr);                                          0183
                    dstptr ← gpstid;                                        0184
                    FIND dstptr ↑ptr;                                       0185
                    RETURN(getids(&ptr, &astr, infotype, idfile));          0186
                    END;                                                    0187
                 END;                                                       0188
              END;                                                          0189
           END;                                                             0190
        %Now edit and append to astr %                                      0191
           IF infotype = 1 THEN                                             0192
              getifnf($infostr,  $infostr);                                 0193
           *astr* ← *astr*, *infostr*;                                      0194
     RETURN(TRUE) END.                                                      0195

%....ident pushdown stack support....%                                      0219
    (intids) %initialize ident pushdown stack%
    PROCEDURE (ptr);                                                        0220
       RESET jidstk;                                                        0221
       IF ptr THEN pushids(ptr);                                            0222
       jidsbot ← jidstk;                                                    0223
       RETURN;                                                              0224
       END.                                                                 0225

    (popids) %pop the ident pushdown stack%
    PROCEDURE (ptr);                                                        0226
       REF ptr;                                                             0227
       IF jidsbot = jidstk THEN RETURN(FALSE);                              0228
       POP jidstk TO ptr;                                                   0229
       RETURN(TRUE);                                                        0230
       END?                                                                 0231

    (pushids) %push the ident pushdown stack%
    PROCEDURE (ptr);                                                        0232
       REF ptr;                                                             0233
       PUSH ptr ON jidstk;                                                  0234
       RETURN;                                                              0235
       END.                                                                 0236

    (gbotids) %collaps ident pushdown stack%
    PROCEDURE (ptr);                                                        0237
       REF ptr;                                                             0238
```

```
    IF jidsbot = jidstk THEN RETURN(FALSE);                              0239
    IF jidstk.systks=1 THEN ptr ← [$jidstk+2]                            0240
    ELSE mvbfbf($jidstk+2, &ptr, jidstk.systks);                         0241
    RETURN(TRUE);                                                        0242
    END.                                                                 0243
                                                                         0244
%....generate status string for an entry in ident-file....%              0810
  (idstatus) %generate status string for an entry%
  PROCEDURE (idstr, entrystr, typid, statstr);                           0811
    %This procedure inserts the status of the entry passed into the
    status string.%                                                      0812
    LOCAL TEXT POINTER ptr1, ptr2, ptr3, ptr4;                           0813
    LOCAL f1, f2;                                                        0814
    LOCAL STRING cadstr[300];                                            0815
    REF idstr, statstr;                                                  0816
    *statstr* ← "Ident: ", *idstr*, EOL;                                 0817
    getinam(entrystr, 0, $ptr1, $ptr2);                                  0818
    *statstr* ← *statstr*, "Name: ", ptr1 ptr2, EOL;                     0819
    IF typid = indtyp THEN                                               0820
       BEGIN                                                             0821
       getiorg(entrystr, 0, $ptr1, $ptr2);                               0822
       *statstr* ← *statstr*, "Organization: ", ptr1 ptr2, EOL;          0823
       IF getisorg(entrystr, 0, $ptr1, $ptr2) THEN                       0824
          *statstr* ← *statstr*, "Secondary Organization(s): ",          0825
             ptr1 ptr2, EOL;                                             0826
       END                                                               0827
    ELSE                                                                 0828
       BEGIN                                                             0829
       IF expdtst(entrystr, 0) THEN                                      0830
          *statstr* ← *statstr*, "Expand", EOL                           0831
       ELSE *statstr* ← *statstr*, "Unexpanded", EOL;                    0832
       getimem(entrystr, 0, $ptr1, $ptr2);                               0833
       *statstr* ← *statstr*, "Membership: ", ptr1 ptr2, EOL;            0834
       geticord(entrystr, 0, $ptr1, $ptr2);                              0835
       *statstr* ← *statstr*, "Coordinator: ", ptr1 ptr2, EOL;           0836
       IF typid = orgtyp THEN                                            0837
          BEGIN                                                          0838
          getityp(entrystr, 0, $ptr1, $ptr2);                            0839
          *statstr* ← *statstr*, "Organization Type: ", ptr1 ptr2,
          EOL;                                                           0840
          END;                                                           0841
       END;                                                              0842
    IF getigrps (entrystr, 0, $ptr1, $ptr2) THEN                         02630
       *statstr* ← *statstr*, "Groups: ", ptr1 ptr2, EOL;                02632
    %Put Online and Network addresses in cadstr%                         0843
       *cadstr* ← NULL;                                                  0844
       f2 ← getinlhost(entrystr, 0, $ptr3, $ptr4);                       0845
       IF (f1 ← getiuser(entrystr, 0, $ptr1, $ptr2)) OR f2 THEN          0846
          BEGIN                                                          0847
          *cadstr* ← *cadstr*, "  Online(NLS)", EOL;                     0848
          IF f1 THEN *cadstr* ← *cadstr*, "    User: ", ptr1 ptr2,
          EOL;                                                           0849
          IF f2 THEN *cadstr* ← *cadstr*, "    Host: ", ptr3 ptr4,
          EOL;                                                           0850
          END;                                                           0851
       f2 ← getihost(entrystr, 0, $ptr3, $ptr4);                         0852
```

```
      IF (f1 ← getinma(entrystr, 0, $ptr1, $ptr2)) OR f2 THEN
         BEGIN
         *cadstr* ← *cadstr*, "  Network", EOL;
         IF f1 THEN *cadstr* ← *cadstr*, "    User: ", ptr1 ptr2,
         EOL;
         IF f2 THEN *cadstr* ← *cadstr*, "    Host: ", ptr3 ptr4,
         EOL;
         END;
   getiadd(entrystr, 0, $ptr1, $ptr2);
   IF cadstr.L > 0 OR ptr2[1] > ptr1[1] THEN
      BEGIN
      *statstr* ← *statstr*, "Mail Addresses: ", EOL;
      IF ptr2[1] > ptr1[1] THEN
         *statstr* ← *statstr*, "  Hardcopy Address: ", ptr1 ptr2,
         EOL;
      *statstr* ← *statstr*, *cadstr*;
      END;
   getiverify(entrystr, 0, $ptr1, $ptr2);
   *statstr* ← *statstr*, ptr1 ptr2, EOL;
   IF getiphone(entrystr, 0, $ptr1, $ptr2) THEN
      BEGIN
      *statstr* ← *statstr*, "Phone: ", ptr1 ptr2, EOL;
      END;
   IF getifunction(entrystr, 0, $ptr1, $ptr2) THEN
      BEGIN
      *statstr* ← *statstr*, "Function: ", ptr1 ptr2, EOL;
      END;
   IF identwheel AND geticapability(entrystr, 0, $ptr1, $ptr2) THEN

      BEGIN
      *statstr* ← *statstr*, "Capabilities: ", ptr1 ptr2, EOL;
      END;
   IF getisubcol(entrystr, 0, $ptr1, $ptr2) THEN
      BEGn
      *statstr* ← *statstr*, "Sub-Collection: ", ptr1 ptr2, EOL;
      END;
   IF getidelivery(entrystr, 0, $ptr1, $ptr2) THEN
      BEGIN
      *statstr* ← *statstr*, "Delivery: ", ptr1 ptr2, EOL;
      END;
   IF getimcmnts(entrystr, 0, $ptr1, $ptr2) THEN
      BEGIN
      *statstr* ← *statstr*, "Comments: ", ptr1 ptr2, EOL;
      END;
   RETURN;
   END.


%....replace fields for an entry in ident-file....%
   (setiid) %set IDENT field for an ident entry%
   PROCEDURE (entrystr, repstr, ptr1, ptr2);
      LOCAL TEXT POINTER lptr1, lptr2;
      REF entrystr, repstr, ptr1, ptr2;
      IF NOT &ptr1 THEN getiid(&entrystr, 0, $lptr1, $lptr2)
      ELSE FIND ptr1 ↑lptr1 ptr2 ↑lptr2;
      *entrystr* ← SF(*entrystr*) lptr1, *repstr*, lptr2 SE(*entrystr*);
```

```
                                                                    0902
   RETURN;                                                          0903
   END.
                                                                    0904
(setinam) %set NAME field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                           0905
   LOCAL TEXT POINTER lptr1, lptr2;                                 0906
   REF entrystr, repstr, ptr1, ptr2;                                0907
   IF NOT &ptr1 THEN getinam(&entrystr, 0, $lptr1, $lptr2)          0908
   ELSE FIND ptr1 ↑lptr1 ptr2 ↑lptr2;                               0909
   *entrystr* ← SF(*entrystr*) lptr1, *repstr*, lptr2 SE(*entrystr*);
                                                                    0910
   RETURN;                                                          0911
   END.
                                                                    0912
(setimem) %set MEMBERSHIP field for a group/organization ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                           0913
   LOCAL STRING mem [500], ident [40];                              02646
   LOCAL TEXT POINTER lptr1, lptr2, lptr3, lptr4;                   0914
   REF entrystr, repstr, ptr1, ptr2;                                0915
   IF NOT &ptr1 THEN getimem(&entrystr, $mem, $lptr1, $lptr2)       0916
   ELSE                                                             0917
      BEGIN                                                         02648
      FIND ptr1 ↑lptr1 ptr2 ↑lptr2;                                 02647
      *mem* ← lptr1 lptr2;                                          02650
      END;                                                          02649
   getiid (&entrystr, $ident, 0, 0);                                02651
   IF *ident* # "NEWIDS" AND *ident* # "DELETE" AND NOT getiprevmem
   (&entrystr, 0, $lptr3, $lptr4) THEN                              02652
      setiprevmem (&entrystr, $mem, $lptr3, $lptr4);                02653
   IF *repstr*[repstr.L] # '; THEN *repstr* ← *repstr*, ';;         0918
   *entrystr* ← SF(*entrystr*) lptr1, *repstr*, lptr2 SE(*entrystr*);
                                                                    0919
   RETURN;                                                          0920
   END.
                                                                    0921
(setiprevmem) %set MEMBERSHIPPREVIOUS field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                           02553
   LOCAL repcnt;                                                    02554
   LOCAL TEXT POINTER p1, p2;                                       02555
   REF entrystr, repstr, ptr1, ptr2;                                02556
   IF NOT &ptr1 THEN getiprevmem(&entrystr, 0, $p1, $p2)            02557
   ELSE FIND ptr1 ↑p1 ptr2 ↑p2;                                     02558
   IF *repstr* = *nullfield* THEN %delete the field%                02559
      IF (FIND p1 < [EOL] > [↑p1 "MembershipPrevious:"] p2 > [';]
      [EOL] ↑p2) THEN                                               02560
         *entrystr* ← SF(*entrystr*) p1, p2 SE(*entrystr*)          02561
      ELSE NULL                                                     02562
   ELSE                                                             02563
      BEGIN                                                         02564
      repcnt ← repstr.L;                                            02565
      IF *repstr* [repstr.L] = '; THEN BUMP DOWN repcnt;            02566
      IF NOT (FIND p1 < [EOL] > ["MembershipPrevious:"]) THEN       02567
         *entrystr* ← SF(*entrystr*) p1, "MembershipPrevious: ",
                                                                    02568
            *repstr* [1 TO repcnt], ';, EOL, p2 SE(*entrystr*)      02569
```

```
      ELSE                                                          02570
         *entrystr* ← SF(*entrystr*) p1, *repstr* [1 TO repcnt],
                                                                    02571
            p2 SE(*entrystr*);                                      02572
      END;                                                          02573
   RETURN;                                                          02574
   END.
                                                                    02575

(setiexp) %set EXPAND field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                           0922
   LOCAL TEXT POINTER lptr1, lptr2;                                 0923
   REF entrystr, repstr, ptr1, ptr2;                                0924
   IF NOT &ptr1 THEN getiexp(&entrystr, 0, $1ptr1, $1ptr2)          0925
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                               0926
   *entrystr* ← SF(*entrystr*) 1ptr1, *repstr*, 1ptr2 SE(*entrystr*);
                                                                    0927
   RETURN;                                                          0928
   END.
                                                                    0929

(seticord) %set COORDINATOR field for a group/organization ident
entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                           0930
   LOCAL TEXT POINTER lptr1, lptr2;                                 0931
   REF entrystr, repstr, ptr1, ptr2;                                0932
   IF NOT &ptr1 THEN geticord(&entrystr, 0, $1ptr1, $1ptr2)         0933
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                               0934
   *entrystr* ← SF(*entrystr*) 1ptr1, *repstr*, 1ptr2 SE(*entrystr*);
                                                                    0935
   RETURN;                                                          0936
   END.
                                                                    0937

(setityp) %set TYPE-OF-ORGANIZATION field for an organization ident
entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                           0938
   LOCAL TEXT POINTER lptr1, lptr2;                                 0939
   REF entrystr, repstr, ptr1, ptr2;                                0940
   IF NOT &ptr1 THEN getityp(&entrystr, 0, $1ptr1, $1ptr2)          0941
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                               0942
   IF NOT (FIND 1ptr1 < [EOL] > ["Type: "]) THEN                    0943
      *entrystr* ← SF(*entrystr*) 1ptr1, "Type:", *repstr*, ';, EOL,
      1ptr2 SE(*entrystr*)                                          0944
   ELSE                                                             0945
      *entrystr* ← SF(*entrystr*) 1ptr1, *repstr*, 1ptr2
      SE(*entrystr*);                                               0946
   RETURN;                                                          0947
   END.
                                                                    0948

(setiadd) %set MAIL ADDRESS field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                           0949
   LOCAL TEXT POINTER lptr1, lptr2;                                 0950
   REF entrystr, repstr, ptr1, ptr2;                                0951
   IF NOT &ptr1 THEN getiadd(&entrystr, 0, $1ptr1, $1ptr2)          0952
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                               0953
   *entrystr* ← SF(*entrystr*) 1ptr1, *repstr*, 1ptr2 SE(*entrystr*);
                                                                    0954
   RETURN;                                                          0955
```

```
   END.                                                                 0956
(setiorg) %set ORGANIZATION field for an individual ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                               0957
   LOCAL TEXT POINTER lptr1, lptr2;                                     0958
   REF entrystr, repstr, ptr1, ptr2;                                    0959
   IF NOT &ptr1 THEN getiorg(&entrystr, 0, $lptr1, $lptr2)              0960
   ELSE FIND ptr1 ↑lptr1 ptr2 ↑lptr2;                                   0961
   *entrystr* ← SF(*entrystr*) lptr1, *repstr*, lptr2 SE(*entrystr*);
                                                                        0962
   RETURN;                                                              0963
   END.
                                                                        0964
(setiuser) %set USER (NLS delivery) field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                               0965
   LOCAL TEXT POINTER lptr1, lptr2;                                     0966
   REF entrystr, repstr, ptr1, ptr2;                                    0967
   IF NOT &ptr1 THEN getiuser(&entrystr, 0, $lptr1, $lptr2)             0968
   ELSE FIND ptr1 ↑lptr1 ptr2 ↑lptr2;                                   0969
   IF *repstr* = *nullfield* THEN %delete the field%                    0970
      IF (FIND lptr1 < [EOL] > [↑lptr1 "User:"] lptr2 > [';] [EOL]
      ↑lptr2) THEN                                                      0971
         *entrystr* ← SF(*entrystr*) lptr1, lptr2 SE(*entrystr*)        0972
      ELSE NULL                                                         0973
   ELSE                                                                 0974
      IF NOT (FIND lptr1 < [EOL] > ["User:"]) THEN                      0975
         *entrystr* ← SF(*entrystr*) lptr1, "User: ",                   0976
            *repstr*, ';, EOL, lptr2 SE(*entrystr*)                     0977
      ELSE                                                              0978
         *entrystr* ← SF(*entrystr*) lptr1, *repstr*,                   0979
            lptr2 SE(*entrystr*);                                       0980
   RETURN;                                                              0981
   END.
                                                                        0982
(setigrps) %set GROUPS field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                               0983
   LOCAL repcnt;                                                        0984
   LOCAL TEXT POINTER lptr1, lptr2;                                     0985
   REF entrystr, repstr, ptr1, ptr2;                                    0986
   IF NOT &ptr1 THEN getigrps(&entrystr, 0, $lptr1, $lptr2)             0987
   ELSE FIND ptr1 ↑lptr1 ptr2 ↑lptr2;                                   0988
   IF *repstr* = *nullfield* THEN %delete the field%                    0989
      IF (FIND lptr1 < [EOL] > [↑lptr1 "Groups:"] lptr2 > [';] [EOL]
      ↑lptr2) THEN                                                      0990
         *entrystr* ← SF(*entrystr*) lptr1, lptr2 SE(*entrystr*)        0991
      ELSE NULL                                                         0992
   ELSE                                                                 0993
      BEGIN                                                             0994
      repcnt ← repstr.L;                                                0995
      IF *repstr* [repstr.L] = '; THEN BUMP DOWN repcnt;                0996
      IF NOT (FIND lptr1 < [EOL] > ["Groups:"]) THEN                    0997
         *entrystr* ← SF(*entrystr*) lptr1, "Groups: ",                 0998
            *repstr* [1 TO repcnt], ';, EOL, lptr2 SE(*entrystr*)
                                                                        0999
      ELSE                                                             01000
         *entrystr* ← SF(*entrystr*) lptr1, *repstr* [1 TO repcnt],
```

```
                                                                    01001
            lptr2 SE(*entrystr*);                                   01002
      END;                                                          01003
   RETURN;                                                          01004
   END.
                                                                    01005
(setihost) %set NET-DELIVERY HOST field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                           01006
   LOCAL TEXT POINTER lptr1, lptr2;                                 01007
   REF entrystr, repstr, ptr1, ptr2;                                01008
   IF NOT &ptr1 THEN getihost(&entrystr, 0, $lptr1, $lptr2)         01009
   ELSE FIND ptr1 ↑lptr1 ptr2 ↑lptr2;                               01010
   IF *repstr* = *nullfield* THEN %delete the field%                01011
      IF (FIND lptr1 < [EOL] > [↑lptr1 "Host:"] lptr2 > [';] [EOL]
      ↑lptr2) THEN                                                  01012
         *entrystr* ← SF(*entrystr*) lptr1, lptr2 SE(*entrystr*)
                                                                    01013
      ELSE NULL                                                     01014
   ELSE                                                             01015
      IF NOT (FIND lptr1 < [EOL] > ["Host:"]) THEN                  01016
         *entrystr* ← SF(*entrystr*) lptr1, "Host: ",               01017
            *repstr*, ';, EOL, lptr2 SE(*entrystr*)                 01018
      ELSE                                                          01019
         *entrystr* ← SF(*entrystr*) lptr1, *repstr*,               01020
            lptr2 SE(*entrystr*);                                   01021
   RETURN;                                                          01022
   END.
                                                                    01023
(setinlhost) %set NLS-DELIVERY HOST field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                           01024
   LOCAL TEXT POINTER lptr1, lptr2;                                 01025
   REF entrystr, repstr, ptr1, ptr2;                                01026
   IF NOT &ptr1 THEN getinlhost(&entrystr, 0, $lptr1, $lptr2)       01027
   ELSE FIND ptr1 ↑lptr1 ptr2 ↑lptr2;                               01028
   IF *repstr* = *nullfield* THEN %delete the field%                01029
      IF (FIND lptr1 < [EOL] > [↑lptr1 "NLS host:"] lptr2 > [';]
      [EOL] ↑lptr2) THEN                                            01030
         *entrystr* ← SF(*entrystr*) lptr1, lptr2 SE(*entrystr*)
                                                                    01031
      ELSE NULL                                                     01032
   ELSE                                                             01033
      IF NOT (FIND lptr1 < [EOL] > ["NLS host:"]) THEN              01034
         *entrystr* ← SF(*entrystr*) lptr1, "NLS host: ",           01035
            *repstr*, ';, EOL, lptr2 SE(*entrystr*)                 01036
      ELSE                                                          01037
         BEGIN                                                      01038
         FIND lptr1 < $NP > $1NP;                                   01039
         ST lptr1 lptr2 ← *repstr*;                                 01040
         END;                                                       01041
   RETURN;                                                          01042
   END.
                                                                    01043
(setinma) %set NETWORK MAILBOX address field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2); %Local Network Mailbox
Address%                                                            01044
   LOCAL TEXT POINTER lptr1, lptr2;                                 01045
```

```
   REF entrystr, repstr, ptr1, ptr2;                                    01046
   IF NOT &ptr1 THEN getinma(&entrystr, 0, $1ptr1, $1ptr2)              01047
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                                   01048
   IF *repstr* = *nullfield* THEN %delete the field%                    01049
      IF (FIND 1ptr1 < [EOL] > [↑1ptr1 "Local Network Mailbox
      Address:"] 1ptr2 > [';] [EOL] ↑1ptr2) THEN                        01050
         *entrystr* ← SF(*entrystr*) 1ptr1, 1ptr2 SE(*entrystr*)
                                                                       01051
      ELSE NULL                                                        01052
   ELSE                                                                01053
      IF NOT (FIND 1ptr1 < [EOL] > ["Local Network Mailbox
      Address:"]) THEN                                                 01054
         *entrystr* ← SF(*entrystr*) 1ptr1, "Local Network Mailbox
         Address: ",                                                   01055
            *repstr*, ';, EOL, 1ptr2 SE(*entrystr*)                    01056
      ELSE                                                             01057
         *entrystr* ← SF(*entrystr*) 1ptr1, *repstr*,                  01058
            1ptr2 SE(*entrystr*);                                      01059
   RETURN;                                                             01060
   END.
                                                                       01061
(setiverify) %set VERYFIED-BY-NIC-PERSONNEL field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                              01062
   LOCAL TEXT POINTER 1ptr1, 1ptr2;                                    01063
   REF entrystr, repstr, ptr1, ptr2;                                   01064
   IF NOT &ptr1 THEN getiverify(&entrystr, 0, $1ptr1, $1ptr2)          01065
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                                  01066
   *entrystr* ← SF(*entrystr*) 1ptr1, *repstr*,                        01067
      1ptr2 SE(*entrystr*);                                            01068
   RETURN;                                                             01069
   END.
                                                                       01070
(setiphone) %set PHONE field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                              01071
   LOCAL TEXT POINTER 1ptr1, 1ptr2;                                    01072
   REF entrystr, repstr, ptr1, ptr2;                                   01073
   IF NOT &ptr1 THEN getiphone(&entrystr, 0, $1ptr1, $1ptr2)           01074
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                                  01075
   IF *repstr* = *nullfield* THEN %delete the field%                   01076
      IF (FIND 1ptr1 < [EOL] > [↑1ptr1 "Phone:"] 1ptr2 > [';] [EOL]
      ↑1ptr2) THEN                                                     01077
         *entrystr* ← SF(*entrystr*) 1ptr1, 1ptr2 SE(*entrystr*)
                                                                       01078
      ELSE NULL                                                        01079
   ELSE                                                                01080
      IF NOT (FIND 1ptr1 < [EOL] > ["Phone:"]) THEN                    01081
         *entrystr* ← SF(*entrystr*) 1ptr1, "Phone: ",                 01082
            *repstr*, ';, EOL, 1ptr2 SE(*entrystr*)                    01083
      ELSE                                                             01084
         *entrystr* ← SF(*entrystr*) 1ptr1, *repstr*,                  01085
            1ptr2 SE(*entrystr*);                                      01086
   RETURN;                                                             01087
   END.
                                                                       01088
(setifunction) %set FUNCTION field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                              01089
```

```
   LOCAL TEXT POINTER lptr1, lptr2;                                    01090
   REF entrystr, repstr, ptr1, ptr2;                                   01091
   IF NOT &ptr1 THEN getifunction(&entrystr, 0, $1ptr1, $1ptr2)        01092
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                                  01093
   IF *repstr* = *nullfield* THEN %delete the field%                   01094
      IF (FIND 1ptr1 < [EOL] > [↑1ptr1 "Function:"] 1ptr2 > [';]
      [EOL] ↑1ptr2) THEN                                               01095
         *entrystr* ← SF(*entrystr*) 1ptr1, 1ptr2 SE(*entrystr*)
                                                                       01096
      ELSE NULL                                                        01097
   ELSE                                                                01098
      IF NOT (FIND 1ptr1 < [EOL] > ["Function:"]) THEN                 01099
         *entrystr* ← SF(*entrystr*) 1ptr1, "Function: ",              01100
            *repstr*, ';, EOL, 1ptr2 SE(*entrystr*)                    01101
      ELSE                                                             01102
         *entrystr* ← SF(*entrystr*) 1ptr1, *repstr*,                  01103
            1ptr2 SE(*entrystr*);                                      01104
   RETURN;                                                             01105
   END.
                                                                       01106
(setisorg) %set SECONDARY ORGANIZATION field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                              01107
   LOCAL TEXT POINTER 1ptr1, 1ptr2;                                    01108
   REF entrystr, repstr, ptr1, ptr2;                                   01109
   IF NOT &ptr1 THEN getisorg(&entrystr, 0, $1ptr1, $1ptr2)            01110
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                                  01111
   IF *repstr* = *nullfield* THEN %delete the field%                   01112
      IF (FIND 1ptr1 < [EOL] > [↑1ptr1 "Secondary organization:"]
      1ptr2 > [';] [EOL] ↑1ptr2) THEN                                  01113
         *entrystr* ← SF(*entrystr*) 1ptr1, 1ptr2 SE(*entrystr*)
                                                                       01114
      ELSE NULL                                                        01115
   ELSE                                                                01116
      IF NOT (FIND 1ptr1 < [EOL] > ["Secondary organization:"]) THEN
                                                                       01117
         *entrystr* ← SF(*entrystr*) 1ptr1, "Secondary organization:
         ",                                                            01118
            *repstr*, ';, EOL, 1ptr2 SE(*entrystr*)                    01119
      ELSE                                                             01120
         *entrystr* ← SF(*entrystr*) 1ptr1, *repstr*,                  01121
            1ptr2 SE(*entrystr*);                                      01122
   RETURN;                                                             01123
   END.
                                                                       01124
(seticapability) %set CAPABILITIES field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                              01125
   LOCAL TEXT POINTER 1ptr1, 1ptr2;                                    01126
   REF entrystr, repstr, ptr1, ptr2;                                   01127
   IF NOT &ptr1 THEN geticapability(&entrystr, 0, $1ptr1, $1ptr2)
                                                                       01128
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                                  01129
   IF *repstr* = *nullfield* THEN %delete the field%                   01130
      IF (FIND 1ptr1 < [EOL] > [↑1ptr1 "Capabilities:"] 1ptr2 > [';]
      [EOL] ↑1ptr2) THEN                                               01131
         *entrystr* ← SF(*entrystr*) 1ptr1, 1ptr2 SE(*entrystr*)
                                                                       01132
```

```
      ELSE NULL                                                    01133
   ELSE                                                            01134
      IF NOT (FIND lptr1 < [EOL] > ["Capabilities:"]) THEN         01135
         *entrystr* ← SF(*entrystr*) lptr1, "Capabilities: ",      01136
            *repstr*, ';, EOL, lptr2 SE(*entrystr*)                01137
      ELSE                                                         01138
         *entrystr* ← SF(*entrystr*) lptr1, *repstr*,              01139
            lptr2 SE(*entrystr*);                                  01140
   RETURN;                                                         01141
   END.                                                            01142

(setisubcol) %set SUBCOLLECTIONS field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                          01143
   LOCAL TEXT POINTER lptr1, lptr2;                                01144
   REF entrystr, repstr, ptr1, ptr2;                               01145
   IF NOT &ptr1 THEN getisubcol(&entrystr, 0, $1ptr1, $1ptr2)      01146
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                              01147
   IF *repstr* = *nullfield* THEN %delete the field%               01148
      IF (FIND lptr1 < [EOL] > [↑1ptr1 "Sub-Collection:"] lptr2 >
      [';] [EOL] ↑1ptr2) THEN                                      01149
         *entrystr* ← SF(*entrystr*) lptr1, lptr2 SE(*entrystr*)
                                                                   01150
      ELSE NULL                                                    01151
   ELSE                                                            01152
      IF NOT (FIND lptr1 < [EOL] > ["Sub-Collection:"]) THEN       01153
         *entrystr* ← SF(*entrystr*) lptr1, "Sub-Collection: ",    01154
            *repstr*, ';, EOL, lptr2 SE(*entrystr*)                01155
      ELSE                                                         01156
         *entrystr* ← SF(*entrystr*) lptr1, *repstr*,              01157
            lptr2 SE(*entrystr*);                                  01158
   RETURN;                                                         01159
   END.                                                            01160

(setidelivery) %set DELIVERY-TYPES field for an ident entry%
PROCEDURE (entrystr, repstr, ptr1, ptr2);                          01161
;\) LPTRGAB TEXT POINTER lptr1, lptr2;                             01162
   REF entrystr, repstr, ptr1, ptr2;                               01163
   IF NOT &ptr1 THEN getidelivery(&entrystr, 0, $1ptr1, $1ptr2)    01164
   ELSE FIND ptr1 ↑1ptr1 ptr2 ↑1ptr2;                              01165
   IF *repstr* = *nullfield* THEN %delete the field%               01166
      IF (FIND lptr1 < [EOL] > [↑1ptr1 "Delivery:"] lptr2 > [';]
      [EOL] ↑1ptr2) THEN                                           01167
         *entrystr* ← SF(*entrystr*) lptr1, lptr2 SE(*entrystr*)
                                                                   01168
      ELSE NULL                                                    01169
   ELSE                                                            01170
      IF NOT (FIND lptr1 < [EOL] > ["Delivery:"]) THEN             01171
         *entrystr* ← SF(*entrystr*) lptr1, "Delivery: ",          01172
            *repstr*, ';, EOL, lptr2 SE(*entrystr*)                01173
      ELSE                                                         01174
         *entrystr* ← SF(*entrystr*) lptr1, *repstr*,              01175
            lptr2 SE(*entrystr*);                                  01176
   RETURN;                                                         01177
   END.                                                            01178

(setimcmnts) %set COMMENTS field for an ident entry%
```

```
   PROCEDURE (entrystr, repstr, ptr1, ptr2);                              01179
      LOCAL TEXT POINTER lptr1, lptr2, tmpp1, tmpp2;                      01180
      REF entrystr, repstr, ptr1, ptr2;                                   01181
      IF NOT &ptr1 THEN getimcmnts(&entrystr, 0, $lptr1, $lptr2)          01182
      ELSE FIND ptr1 ↑lptr1 ptr2 ↑lptr2;                                  01183
      IF *repstr* = *nullfield* THEN %delete the field%                   01184
         IF (FIND lptr1 < [EOL] > [↑lptr1 "Comments:"] lptr2 > [EOL]
         ↑lptr2) THEN                                                     01185
            *entrystr* ← SF(*entrystr*) lptr1, lptr2 SE(*entrystr*)
                                                                          01186
         ELSE NULL                                                        01187
      ELSE                                                                01188
         IF NOT (FIND lptr1 < [EOL] > ["Comments:"]) THEN                 01189
            *entYSTR* ← SF(*entrystr*) lptr1, "Comments: ",              01190
               *repstr*, EOL, lptr2 SE(*entrystr*)                        01191
         ELSE                                                             01192
            *entrystr* ← SF(*entrystr*) lptr1, *repstr*,                  01193
               lptr2 SE(*entrystr*);                                      01194
      RETURN;                                                             01195
      END.
                                                                          01196
%....update ident file....%                                               01197
   (upidfil) %update add a new record to the master ident-file%
   PROCEDURE (idstr, infostr, fileno, doit);                              01198
      %This procedure uses xupidfil to add new ident records to the
      ident file.  If the new id is already in the file, it doesn't
      update, and returns FALSE.  Otherwise it returns TRUE.%             01199
      %if doit then updates the file, otherwise closes pc file  --
      leaves file locked but this user does not see pc changes until
      someone updates the file%                                           01200
      LOCAL  idstid, retvalue, trap, capsav;                              01201
      REF idstr, infostr;                                                 01202
      trap ← FALSE;                                                       01203
      capsav ← 0;                                                         02726
      %lock ident file%                                                   01204
         conidir(TRUE);                                                   01205
         idstid ← orgstid;                                                01206
         ON SIGNAL ELSE                                                   01207
            BEGIN                                                         01208
            IF trap THEN notrapcc(capsav);                                01209
            sigclose(idstid.stfile := 0);                                 01210
            conidir(FALSE);                                               01211
            END;                                                          01212
         idstid ← openlock(0, jflname($"identfile"));                     01213
         enablaccess(idstid.stfile, jrnlaccess);                          01214
      capsav ← trapcc();                                                  01215
      trap ← TRUE;                                                        01216
      retvalue ← xupidfil(&idstr, &infostr, idstid.stfile) ;              01217
      %update file, connect back to original directory%                   01218
         IF idstid.stfile NOT= fileno THEN                                01219
            BEGIN                                                         01220
            IF doit AND retvalue THEN                                     01221
               closeu(idstid.stfile := 0)                                 01222
            ELSE                                                          01223
               close(idstid.stfile := 0)                                  01224
            END                                                           01225
```

```
      ELSE                                                              01226
         BEGIN                                                          01227
         IF doit AND retvalue THEN                                      01228
            updtfl(idstid.stfile, 1,0)                                  01229
         ELSE                                                           01230
            BEGIN                                                       01231
            closepc(idstid.stfile);                                     01232
            disabaccess(idstid.stfile, jrnlaccess);                     01233
            END;                                                        01234
         END;                                                           01235
      conidir(FALSE);                                                   01236
   notrapcc(capsav);                                                    01237
   trap ← FALSE;                                                        01238
   RETURN(retvalue);                                                    01239
   END.                                                                 01240

(xupidfil) %support routine for UPIDFIL%
PROCEDURE (idstr, infostr, fileno);                                     01241
                                                                        01242
   %This procedure adds new ident records to the ident file.  If the
   new id is already in the file, it doesn't update, and returns
   FALSE.  Otherwise it returns TRUE.  In addition, it add the new
   ident to th "group" of new idents in the ident file, and, if the
   new ident is for an individual, it adds it to the membership list
   of his organization.%                                                01243
   %assumes identfile open and modifiable!!%                            01244
                                                                        01245
   LOCAL  stid, idstid, repstid, i, entry, trap, capsav;                01246
   LOCAL TEXT POINTER z1, z2, z3, z4;                                   01247
   LOCAL STRING memlst[350], orgidstr[150];                             01248
   REF idstr, infostr, entry;                                           01249
   &entry ← getstring (2000, $dspblk);                                  02654
   trap ← FALSE;                                                        02730
   capsav ← 0;                                                          02731
   ON SIGNAL ELSE                                                       02655
      BEGIN                                                             02729
      ON SIGNAL ELSE;                                                   02732
      IF &entry THEN freestring ((&entry := 0), $dspblk);               02728
      IF trap THEN notrapcc(capsav);                                    02733
      END;                                                              02734
   idstid ← orgstid;                                                    01250
   idstid.stfile ← fileno;                                              01251
   %one last check, to make sure newly assigned ident hasn't been
   added in last few minutes%                                           01252
      IF oldid(&idstr, idstid.stfile) THEN                              01253
         RETURN(FALSE);                                                 01254
   capsav ← trapcc();                                                   01255
   trap ← TRUE;                                                         02727
   %add new ident to file%                                              01256
      IF  orgtst(&infostr, 0) THEN                                      01257
         IF (stid ← namelook(idstid, $"'organizations'")) =endfil
         THEN                                                           01258
            stid ← idstid                                               01259
         ELSE NULL                                                      01260
      ELSE                                                              01261
         IF  jgrptst(&infostr, 0) THEN                                  01262
```

```
      IF (stid ← namelook(idstid, $"'groups'")) = endfil THEN
                                                                    01263
         stid ← idstid                                              01264
      ELSE NULL                                                     01265
   ELSE                                                             01266
      BEGIN                                                         01267
      getilname(&infostr, &entry,0,0);                              01268
      FOR i ← 1 UP UNTIL > entry.L DO %convert spaces to ''%
                                                                    01269
         IF *entry*[i] = SP THEN *entry*[i] ← '';                   01270
      *entry* ← '', *entry*, '';                                    01271
      IF (stid ← namelook(idstid, &entry)) = endfil OR NOT
      (FIND SF(stid) [')] $SP "LAST NAME") THEN                     01272
         BEGIN                                                      01273
         *entry* ← '(, *entry*, ") LAST NAME";                      01274
         IF (stid ← namelook(idstid, $"'individuals'")) =
         endfil THEN                                                01275
            stid ← idstid;                                          01276
         FIND SF(*entry*) ↑z1 SE(*entry*) ↑z2;                      01277
         stid ← cinssta(stid, levdown, $z1, $z2);                   01278
         END;                                                       01279
      END;                                                          01280
   upfchgrps (&infostr, idstid.stfile);                             02656
   FIND SF(*infostr*) ↑z1 SE(*infostr*) ↑z2;                        01281
   cinssta(stid, levdown, $z1, $z2);                                01282
   addid(&idstr, idstid.stfile);                                    01283
%add ident to membership list of organization%                      01284
   IF NOT orgrptst(&infostr, 0) THEN                                01285
      BEGIN                                                         01286
      getiorg(&infostr, $orgidstr, 0, 0);                           01287
      IF ckident($orgidstr, &entry, fileno : repstid) THEN          01288
         BEGIN                                                      01289
         getimem(&entry, $memlst, 0, 0);                            01290
         IF memlst.M > memlst.L + idstr.L + 1 THEN                  01291
            BEGIN                                                   01292
            *memlst* ← *idstr*, SP, *memlst*;                       01293
            setimem(&entry, $memlst, 0, 0);                         01294
            upfchgrps (&entry, idstid.stfile);                      02657
            FIND SF(repstid) ↑z1 SE(repstid) ↑z2;                   01295
            FIND SF(*entry*) ↑z3 SE(*entry*) ↑z4;                   01296
            creptex($z1, $z2, $z3, $z4);                            01297
            END;                                                    01298
         END;                                                       01299
      END;                                                          01300
%update list of unverified idents%                                  01301
   IF NOT ckident($nwidstr, &entry, fileno : repstid) THEN          01302
      %group for new ids not in ident file, add it%                 01303
      BEGIN                                                         01304
      *entry* ← '(, *nwidstr*, ") Group", EOL, *idstr*, SP,         01305
         '(, *initsr*, '), ';, EOL, "New Idents", EOL,              01306
         "BER", EOL, "BER", EOL, EOL;                               01307
      IF (stid ← namelook(idstid, $"'groups'")) NOT= endfil THEN
                                                                    01308
         BEGIN                                                      01309
         FIND SF(*entry*) ↑z1 SE(*entry*) ↑z2;                      01310
         cinssta(stid, levdown, $z1, $z2);                          01311
```

```
            END;                                                    01312
         END                                                        01313
      ELSE                                                          01314
         BEGIN                                                      01315
         getimem(&entry, $memlst, 0, 0);                            01316
         IF memlst.M > idstr.L + initsr.L +memlst.L + 4 THEN        01317
            BEGIN                                                   01318
            *memlst* ← *idstr*, SP, '(, *initsr*, '), SP, *memlst*;
                                                                    01319
            setimem(&entry, $memlst, 0, 0);                         01320
            setiprevmem (&entry, $nullfield, 0, 0);                 02658
            FIND SF(repstid) ↑z1 SE(repstid) ↑z2;                   01321
            FIND SF(*entry*) ↑z3 SG(*entry*) ↑z4;                   01322
            creptex($z1, $z2, $z3, $z4);                            01323
            END;                                                    01324
         END;                                                       01325
   IF trap := FALSE THEN notrapcc(capsav);                          01326
   freestring ((&entry := 0), $dsppblk);                            02659
   RETURN(TRUE);                                                    01327
   END.
                                                                    01328
(addid) %add new ident to usedids branch%
PROCEDURE (idstr, fileno);                                          01329
   %fileno must be supplied%                                        01330
   %assumes identfile open and writable%                            01331
   LOCAL idstid,  i, stid, trap, capsav;                            01332
   LOCAL TEXT POINTER z1, z2;                                       01333
   LOCAL STRING numsr[10];                                          01334
   REF idstr;                                                       01335
   idstid ← orgstid;                                                01336
   idstid.stfile ← fileno;                                          01337
   trap ← FALSE;                                                    02735
   capsav ← 0;                                                      02736
   ON SIGNAL ELSE IF trap THEN notrapcc(capsav);                    02737
   IF (stid ← namelook(idstid, $"'usedids'")) NOT= endfil AND (stid
   := getsub(stid)) NOT= stid THEN                                  01338
      BEGIN                                                         01339
      capsav ← trapcc();                                            01340
      trap ← TRUE;                                                  02738
      FIND SF(stid) $NP ↑z1 $D ↑z2;                                 01341
      *numsr* ← z1 z2;                                              01342
      i ← VALUE($numsr);                                            01343
      IF (i ← i+1) > 200 THEN                                       01344
         BEGIN                                                      01345
         *numsr* ← ',;                                              01346
         FIND SF(*numsr*) ↑z1 SE(*numsr*) ↑z2;                      01347
         stid ← cinssta(getup(stid), levdown, $z1, $z2);            01348
         FIND SF(stid) ↑z1 ↑z2;                                     01349
         i ← 1;                                                     01350
         END;                                                       01351
      *numsr* ← STRING(i);                                          01352
      ST stid ← *numsr*, z2 SE(stid), *idstr*, ',;                  01353
      IF trap := FALSE THEN notrapcc(capsav);                       01354
      END;                                                          01355
   IF NOT fileno THEN close(idstid.stfile := 0);                    01356
   RETURN;                                                          01357
```

```
   END.                                                                 01358
(modidfil) %modify old record in the master ident-file%
PROCEDURE (idstr, infostr, fileno, doit);                               01359
   %if doit then update the file, otherwise just close the pc%          01360
   LOCAL i, stid, idstid, repstid, trap, capsav;                        01361
   LOCAL TEXT POINTER z1,z2,z3,z4;                                      01362
   LOCAL STRING work[50];                                               01363
   REF idstr, infostr;                                                  01364
   trap ← FALSE;                                                        01365
   capsav ← 0;                                                          02739
   conidir(TRUE);                                                       01366
   idstid ← orgstid;                                                    01367
   ON SIGNAL ELSE                                                       01368
      BEGIN                                                             01369
      IF trap:= FALSE THEN notrapcc(capsav);                            01370
      sigclose(idstid.stfile := 0);                                     01371
      conidir(FALSE);                                                   01372
      END;                                                              01373
   %First build statement in astr%                                      01374
   idstid ← openlock(0, jflname($"identfile"));                         01375
   enablaccess(idstid.stfile, jrnlaccess);                              01376
   IF ckident(&idstr, 0, idstid.stfile : repstid) THEN                  01377
      BEGIN                                                             01378
      trap ← TRUE;                                                      01379
      capsav ← trapcc();                                                01380
      upfchgrps (&infostr, idstid.stfile);                              02633
      getiid(&infostr, $work,0,0);                                      01381
      IF *work* # *idstr* THEN                                          01382
         modrefs(&idstr, $work, &infostr, idstid.stfile);               01383
      IF NOT orgrptst(&infostr, 0) THEN %an individual%                 01384
         BEGIN                                                          01385
         getilname(&infostr, $work, 0,0);                               01386
         FOR i ← 1 UP UNTIL > work.L DO                                 01387
            IF *work*[i] = SP THEN *work*[i] ← '';                      01388
         *work* ← '', *work*, '';                                       01389
         stid ← getup(repstid);                                         01390
         IF NOT FIND SF(stid) $SP '( $SP *work* $SP ') $SP "LAST
         NAME" THEN %last name changed%                                 01391
            BEGIN                                                       01392
            cdelsta(repstid, FALSE, 0);                                 01393
            IF (stid := getsub(stid)) = stid THEN %no others with old
            last name%                                                  01394
               cdelsta(stid, FALSE, 0);                                 01395
            IF (stid ← namelook(idstid, $work)) = endfil OR NOT FIND
            SF(stid) [')] $SP "LAST NAME" THEN                          01396
               BEGIN                                                    01397
               *work* ← '(, *work*, ") LAST NAME";                      01398
               IF (stid ← namelook(idstid, $"'individuals'")) =
               endfil THEN                                              01399
                  stid ← idstid;                                        01400
               FIND SF(*work*) ↑z1 SE(*work*) ↑z2;                      01401
               stid ← cinssta(stid, levdown, $z1, $z2);                 01402
               END;                                                     01403
            FIND SF(*infostr*) ↑z1 SE(*infostr*) ↑z2;                   01404
            cinssta(stid, levdown, $z1, $z2);                           01405
            END                                                         01406
```

```
      ELSE                                                          01407
         BEGIN                                                      01408
         FIND SF(repstid) ↑z1 SE(repstid) ↑z2;                      01409
         FIND SF(*infostr*) ↑z3 SE(*infostr*) ↑z4;                  01410
         creptex($z1, $z2, $z3, $z4);                               01411
         END;                                                       01412
      END                                                           01413
   ELSE                                                             01414
      BEGIN                                                         01415
      FIND SF(repstid) ↑z1 SE(repstid) ↑z2;                         01416
      FIND SF(*infostr*) ↑z3 SE(*infostr*) ↑z4;                     01417
      creptex($z1, $z2, $z3, $z4);                                  01418
      END;                                                          01419
   IF idstid.stfile = fileno THEN                                   01420
      IF doit THEN                                                  01421
         updtfl(idstid.stfile := 0, 1, 0)                           01422
      ELSE                                                          01423
         BEGIN                                                      01424
         closepc(idstid.stfile);                                    01425
         disabaccess(idstid.stfile := 0, jrnlaccess);               01426
         END                                                        01427
   ELSE                                                             01428
      IF doit THEN                                                  01429
         closeu(idstid.stfile := 0)                                 01430
      ELSE                                                          01431
         close(idstid.stfile := 0);                                 01432
   IF trap:=FALSE THEN notrapcc(capsav);                            01433
   conidir(FALSE);                                                  01435
   RETURN(TRUE);                                                    01436
   END;                                                             01437
close(idstid.stfile :=0);                                           01438
conidir(FALSE);                                                     01439
RETURN(FALSE);                                                      01440
END.                                                                01441

(modrefs) %change all referrences to old-ident to new-ident%
PROCEDURE (oidstr, nidstr, entrystr, fileno);                       01442
   %assume file open fo mods-- dont close it or update it!!!!%      01443
   LOCAL trap,capsav, idstid, repstid, typid, updflg, entry;        01444
   LOCAL TEXT POINTER rptr1, rptr2, z1, z2, z3, z4;                 01445
   LOCAL STRING repstr[500];                                        01446
   REF entry;                                                       02634
   &entry ← getstring (2000, $dspblk);                              02635
   trap ← FALSE;                                                    01447
   capsav ← 0;                                                      02740
   ON SIGNAL ELSE                                                   02636
      BEGIN                                                         02741
      IF trap:=FALSE THEN notrapcc(capsav);                         02743
      IF &entry THEN freestring ((&entry := 0), $dspblk);           02637
      END;                                                          02742
   conidir(TRUE);                                                   01448
   idstid ← orgstid;                                                01449
   idstid.stfile ← fileno;                                          01450
   updflg ← FALSE;                                                  01451
   idstid ← getsub(idstid);                                         01452
   trap ← TRUE;                                                     01453
```

```
   capsav ← trapcc();                                                    01454
   LOOP                                                                  01455
      BEGIN                                                              01456
      IF FIND SF(idstid) $SP '( $SP "'usedids'" $SP ') THEN              01457
         BEGIN                                                           01458
         IF (idstid := getsub(idstid)) NOT= idstid THEN                  01459
            LOOP                                                         01460
               BEGIN                                                     01461
               IF FIND SF(idstid) [', *[oidstr]* ',] < CH ↑rptr2 [',]
               > CH ↑rptr1 THEN                                          01462
                  BEGIN                                                  01463
                  ST idstid ← SF(idstid) rptr1, *[nidstr]*, rptr2
                  SE(idstid);                                            01464
                  idstid ← getup(idstid);                                01465
                  EXIT LOOP;                                             01466
                  END;                                                   01467
               IF getftl(idstid) THEN                                    01468
                  BEGIN                                                  01469
                  idstid ← getsuc(idstid);                               01470
                  EXIT LOOP;                                             01471
                  END;                                                   01472
               idstid ← getsuc(idstid);                                  01473
               END;                                                      01474
         idstid ← getsuc(idstid);                                        01475
         REPEAT LOOP;                                                    01476
         END;                                                            01477
      IF NOT (FIND SF(idstid)
       ( $SP '( $SP (
      "'individuals'"/"'groups'"/"'organizations'"/"help") $SP ') /
       [')] $SP "LAST NAME")) AND (FIND SF(idstid) [*[oidstr]*]) THEN
                                                                        01478
         BEGIN                                                           01479
         *entry* ← SF(idstid) SE(idstid);                                01480
         IF orgrptst(&entry, 0) THEN                                     01481
            BEGIN                                                        01482
            getimem(&entry, $repstr, $rptr1, $rptr2);                    01483
            IF subidstr($repstr, oidstr, nidstr) THEN                    01484
               BEGIN                                                     01485
               updflg ← TRUE;                                            01486
               setimem(&entry, $repstr, $rptr1, $rptr2);                 01487
               setiprevmem (&entry, $nullfield, 0, 0);                   02638
               END;                                                      01488
            getiadd(&entry, $repstr, $rptr1, $rptr2);                    01489
            IF subidstr($repstr, oidstr, nidstr) THEN                    01490
               BEGIN                                                     01491
               updflg ← TRUE;                                            01492
               setiadd(&entry, $repstr, $rptr1, $rptr2);                 01493
               END;                                                      01494
            geticord(&entry, $repstr, $rptr1, $rptr2);                   01495
            IF subidstr($repstr, oidstr, nidstr) THEN                    01496
               BEGIN                                                     01497
               updflg ← TRUE;                                            01498
               seticord(&entry, $repstr, $rptr1, $rptr2);                01499
               END;                                                      01500
            END                                                          01501
         ELSE                                                            01502
```

```
            BEGIN                                                    01503
            getiorg(&entry, $repstr, $rptr1, $rptr2);                01504
            IF subidstr($repstr, oidstr, nidstr) THEN                01505
               BEGIN                                                 01506
               updflg ← TRUE;                                        01507
               setiorg(&entry, $repstr, $rptr1, $rptr2);             01508
               END;                                                  01509
            getiadd(&entry, $repstr, $rptr1, $rptr2);                01510
            IF subidstr($repstr, oidstr, nidstr) THEN                01511
               BEGIN                                                 01512
               updflg ← TRUE;                                        01513
               setiadd(&entry, $repstr, $rptr1, $rptr2);             01514
               END;                                                  01515
            END;                                                     01516
         getigrps (&entry, $repstr, $rptr1, $rptr2);                 02639
         IF subidstr ($repstr, oidstr, nidstr) THEN                  02640
            BEGIN                                                    02641
            updflg ← TRUE;                                           02642
            setigrps (&entry, $repstr, $rptr1, $rptr2);              02643
            END;                                                     02644
         IF updflg := FALSE THEN                                     01517
            BEGIN                                                    01518
            FIND SF(idstid) ↑z1 SE(idstid) ↑z2;                      01519
            FIND SF(*entry*) ↑z3 SE(*entry*) ↑z4;                    01520
            creptex($z1, $z2, $z3, $z4);                             01521
            END;                                                     01522
         END;                                                        01523
      IF (idstid ← getnxt(idstid)) = endfil THEN EXIT LOOP;          01524
      END;                                                           01525
   IF trap:=FALSE THEN notrapcc(capsav);                             01526
   freestring ((&entry := 0), $dspblk);                              02645
   RETURN;                                                           01528
   END.
                                                                     01529
(subidstr) %support routine for MODREFS%
PROCEDURE (string, oidstr, nidstr);                                  01530
   LOCAL TEXT POINTER strtptr, tptr1, tptr2;                         01531
   LOCAL STRING tststr[20];                                          01532
   LOCAL modflg;                                                     01533
   REF string, nidstr, oidstr;                                       01534
   modflg ← FALSE;                                                   01535
   FIND SF(*string*) ↑strtptr;                                       01536
   WHILE                                                             01537
      (FIND strtptr > [(LD/'-)] ↑tptr1 ←tptr1) DO                    01538
         BEGIN                                                       01539
         IF NOT (FIND tptr1 [(NP/';/'(/'))] ↑tptr2 ←tptr2 ↑strtptr)
         THEN                                                        01540
            FIND SE(*string*) ↑tptr2 ↑strtptr;                       01541
         *tststr* ← tptr1 tptr2;                                     01542
         IF *tststr* = *oidstr* THEN                                 01543
            BEGIN                                                    01544
            modflg ← TRUE;                                           01545
            *string* ← SF(*string*) tptr1, *nidstr*, tptr2
            SE(*string*);                                            01546
            END;                                                     01547
         END;                                                        01548
```

```
   RETURN(modflg);                                                    01549
   END.
                                                                      01550
(upfchgrps) %propagate membership change to other idents%
PROCEDURE (entry, openidfileno);                                      02576
   %----------%                                                       02577
   LOCAL auxentry, identstid;                                         02578
   LOCAL STRING ident [40], oldmem [500], newmem [500], memident
   [40], grps [500];                                                  02579
   LOCAL TEXT POINTER m1, m2, p1, p2, g1, g2;                         02580
   REF entry;                                                         02581
   %----------%                                                       02582
   %exit if no change in membership%                                  02583
      IF NOT getiprevmem (&entry, $oldmem, $m1, $m2) THEN             02584
         RETURN;                                                      02585
      setiprevmem (&entry, $nullfield, $m1, $m2);                     02586
   %initializaton%                                                    02587
      auxentry ← getstring (2000, $dspblk);                           02588
      ON SIGNAL ELSE                                                  02589
         IF auxentry THEN                                             02590
            freestring ((auxentry := 0), $dspblk);                    02591
      getiid (&entry, $ident, 0, 0);                                  02592
   %compare old and new memberships%                                  02593
      getimem (&entry, $newmem, 0, 0);                                02594
      delcmidents ($oldmem, TRUE, $newmem, TRUE);                     02595
   %update entries of idents deleted from membership list%            02596
      FIND SF(*oldmem*) ↑p2;                                          02597
      WHILE (FIND p2 > $(SP/',) ↑p1 1$(LD/'-) ↑p2) DO                 02598
         BEGIN                                                        02599
         *memident* ← p1 p2;                                          02600
         IF NOT ckident ($memident, auxentry, openidfileno:
         identstid)                                                   02601
            THEN REPEAT LOOP;                                         02602
         IF getigrps (auxentry, $grps, $g1, $g2) THEN                 02603
            BEGIN                                                     02604
            delcmidents ($ident, FALSE, $grps, TRUE);                 02605
            setigrps (auxentry, $grps, $g1, $g2);                     02606
            crepsta (0, identstid, identstid, auxentry);              02607
            END;                                                      02608
         END;                                                         02609
   %update entries of idents added to membership list%                02610
      FIND SF(*newmem*) ↑p2;                                          02611
      WHILE (FIND p2 > $(SP/',) ↑p1 1$(LD/'-) ↑p2) DO                 02612
         BEGIN                                                        02613
         *memident* ← p1 p2;                                          02614
         IF NOT ckident ($memident, auxentry, openidfileno:
         identstid)                                                   02615
            THEN REPEAT LOOP;                                         02616
         IF getigrps (auxentry, $grps, $g1, $g2) THEN                 02617
            *grps* ← *grps*, SP, *ident*                              02618
         ELSE *grps* ← *ident*;                                       02619
         setigrps (auxentry, $grps, $g1, $g2);                        02620
         crepsta (0, identstid, identstid, auxentry);                 02621
         END;                                                         02622
   %termination%                                                      02623
      freestring ((auxentry := 0), $dspblk);                          02624
```

```
      RETURN;                                                          02625
      END.                                                             02626
                                                                       02627
%....verify ident-file routines....%                                   01846
   (veridfile) %verify Master Ident-file%
   PROCEDURE (fileno, which, contonerr, idstr);                        01847
      LOCAL save, idstid, stid;                                        01848
      LOCAL STRING work[1000];                                         01849
      REF idstr;                                                       01850
      save ← rubabt := TRUE;                                           01851
      idstid ← orgstid;                                                01852
      IF NOT fileno THEN                                               01853
         BEGIN                                                         01854
         ON SIGNAL ELSE close(idstid.stfile := 0);                     01855
         idstid.stfile ← open(0, jflname($"identfile"));               01856
         END                                                           01857
      ELSE                                                             01858
         idstid.stfile ← fileno;                                       01859
      IF which .A filver THEN vfmain(idstid.stfile, TRUE); %file verify%
                                                                       01860
      IF which .A usedids THEN verusedids(idstid.stfile, contonerr,
      &idstr);                                                         01861
      IF idstr.L = empty THEN                                          01862
         BEGIN                                                         01863
         IF which .A inds THEN                                         01864
            verinds(idstid.stfile, contonerr, 0);                      01865
         IF which .A groups THEN                                       01866
            vergroups(idstid.stfile, contonerr, 0, NOT (which .V
            usedids));                                                 01867
         IF which .A orgs THEN                                         01868
            verorgs(idstid.stfile, contonerr, 0, NOT (which .V
            usedids));                                                 01869
         END                                                           01870
      ELSE                                                             01871
         BEGIN                                                         01872
         IF NOT ckident(&idstr, $work, idstid.stfile : stid) THEN      01873
            BEGIN                                                      01874
            typeas($"
            CKIDENT failed for ");                                     01875
            typeas($idstr);                                            01876
            crlf();                                                    01877
            IF NOT contonerr THEN RETURN(FALSE);                       01878
            END                                                        01879
         ELSE                                                          01880
            BEGIN                                                      01881
            IF jgrptst($work,0) THEN                                   01882
               vergroups(idstid.stfile, contonerr, stid, TRUE)         01883
            ELSE                                                       01884
               IF orgtst($work,0) THEN                                 01885
                  verorgs(idstid.stfile, contonerr, stid, TRUE)        01886
               ELSE                                                    01887
                  verinds(idstid.stfile, contonerr, stid);             01888
            END;                                                       01889
         END;                                                          01890
      IF NOT fileno THEN                                               01891
         close(idstid.stfile := 0);                                    01892
```

```
   rubabt ← save;                                                    01893
   RETURN(TRUE);                                                     01894
   END.                                                              01895
                                                                     01896
(verusedids) %verify 'usedids' branch%
PROCEDURE (fileno, contonerr, identsr);                              01897
   LOCAL  i, idstid, stid;                                           01898
   LOCAL TEXT POINTER z1,z2;                                         01899
   LOCAL STRING idstr[50], numsr[50], work[1000];                    01900
   REF identsr;                                                      01901
   typeas($"
   Verifying 'USEDIDS':
   ");                                                               01902
   idstid ← orgstid;                                                 01903
   IF NOT fileno THEN                                                01904
      BEGIN                                                          01905
      ON SIGNAL ELSE close(idstid.stfile := 0);                      01906
      idstid.stfile ← open(0, jflname($"identfile"));                01907
      END                                                            01908
   ELSE                                                              01909
      idstid.stfile ← fileno;                                        01910
   IF identsr.L THEN                                                 01911
      BEGIN                                                          01912
      IF NOT oldid(&identsr, idstid.stfile) THEN                     01913
         BEGIN                                                       01914
         typeas($"
         OLDID failed for ");                                        01915
         typeas($identsr);                                           01916
         crlf();                                                     01917
         IF NOT contonerr THEN RETURN(FALSE);                        01918
         END;                                                        01919
      IF NOT ckident($identsr, 0, idstid.stfile) THEN                01920
         BEGIN                                                       01921
         typeas($"
         CKIDENT failed for ");                                      01922
         typeas($identsr);                                           01923
         crlf();                                                     01924
         IF NOT contonerr THEN RETURN(FALSE);                        01925
         END;                                                        01926
      RETURN(TRUE);                                                  01927
      END;                                                           01928
   IF (stid ← namelook(idstid, $"'usedids'")) NOT= endfil AND (stid
   := getsub(stid)) NOT= stid THEN                                   01929
      LOOP                                                           01930
         BEGIN                                                       01931
         IF FIND SF(stid) $SP ↑z1 $D ↑z2 THEN                        01932
            BEGIN                                                    01933
            *numsr* ← z1 z2;                                         01934
            FOR i ← VALUE($numsr) DOWN UNTIL <= 0 DO                 01935
               IF FIND z2  ', ↑z1 1$(LD/'-) ↑z2 THEN                 01936
                  BEGIN                                              01937
                  *idstr* ← z1 z2;                                   01938
                  IF NOT ckident($idstr, 0, stid.stfile) THEN        01939
                     BEGIN                                           01940
                     typeas($"
                     CKIDENT failed for ");                          01941
```

```
                typeas($idstr);                                    01942
                crlf¤[]:
                if not contonerr THEN RETURN(FALSE);               01944
                END                                                01945
             END                                                   01946
          ELSE %count was too high%                                01947
             BEGIN                                                 01948
             typeas($"
             Count should be ");                                   01949
             *numsr* ← STRING(VALUE($numsr)-i);                    01950
             typeas($numsr);                                       01951
             typeas($" in:  ");                                    01952
             *work* ← SF(stid) z2;                                 01953
             typeas($work);                                        01954
             typeas($"><LF><");                                    01955
             *work* ← z2 SE(stid);                                 01956
             typeas($work);                                        01957
             crlf();                                               01958
             IF NOT contonerr THEN RETURN(FALSE);                  01959
             EXIT;                                                 01960
             END;                                                  01961
       IF FIND z2 ',  1$(LD/'-) THEN %count was low%               01962
          BEGIN                                                    01963
          typeas($"
          count was LOW in:  ");                                   01964
          *work* ← SF(stid) z2;                                    01965
          typeas($work);                                           01966
          typeas($"><LF><");                                       01967
          *work* ← z2 SE(stid);                                    01968
          typeas($work);                                           01969
          crlf();                                                  01970
          IF NOT contonerr THEN RETURN(FALSE);                     01971
          LOOP                                                     01972
             IF FIND z2  ', ↑z1 1$(LD/'-) ↑z2 THEN                 01973
                BEGIN                                              01974
                *idstr* ← z1 z2;                                   01975
                IF NOT ckident($idstr, 0, stid.stfile) THEN
                                                                   01976
                   BEGIN                                           01977
                   typeas($"
                   CKIDENT failed for ");                          01978
                   typeas($idstr);                                 01979
                   crlf();                                         01980
                   IF NOT contonerr THEN RETURN(FALSE);            01981
                   END                                             01982
                END                                                01983
             ELSE GXIT;                                            01984
          END;                                                     01985
       END                                                         01986
    ELSE                                                           01987
       BEGIN                                                       01988
       typeas($"
       Count missing in:  ");                                      01989
       *work* ← SF(stid) SE(stid);                                 01990
       typeas($work);                                              01991
       crlf();                                                     01992
```

```
            IF NOT contonerr THEN RETURN(FALSE);                    01993
            END;                                                    01994
         IF getft1(stid) THEN EXIT;                                 01995
         stid ← getsuc(stid);                                       01996
         END                                                        01997
   ELSE                                                             01998
      IF stid = endfil THEN                                         01999
         typeas($"
         No 'USEDIDS' branch
         ")                                                         02000
      ELSE                                                          02001
         typeas($"
         'USEDIDS' branch has no substructure
         ");                                                        02002
   IF NOT fileno THEN                                               02003
      close(idstid.stfile := 0);                                    02004
   RETURN(TRUE);                                                    02005
   END.                                                             02006
                                                                    02007
(verinds) %verify 'individuals' branch%
PROCEDURE (fileno, contonerr, identstid);                           02008
   %consistency check on last name, oldid, ckident%                 02009
   LOCAL idstid, rstid, stid;                                       02010
   LOCAL TEXT POINTER z1,z2,z3;                                     02011
   LOCAL STRING                                                     02012
      cntsr[10], work[1000], idstr[50], uplname[50];                02013
   typeas($"
   Verifying 'INDIVIDUALS':
   ");                                                              02014
   idstid ← orgstid;                                                02015
   IF NOT fileno THEN                                               02016
      BEGIN                                                         02017
      ON SIGNAL ELSE close(idstid.stfile := 0);                     02018
      idstid.stfile ← open(0, jflname($"identfile"));               02019
      END                                                           02020
   ELSE                                                             02021
      idstid.stfile ← fileno;                                       02022
   IF identstid THEN %process a single ident%                       02023
      BEGIN                                                         02024
      stid ← getup(identstid);                                      02025
      IF FIND SF(stid) $SP '( $SP '' ↑z1 [')] < CH $SP '' > ↑z2 [')]
      $SP "LAST NAME" THEN                                          02026
         BEGIN                                                      02027
         *uplname* ← z1 z2;                                         02028
         IF NOT xvinds(identstid, $uplname, contonerr) THEN
         RETURN(FALSE);                                             02029
         END                                                        02030
      ELSE                                                          02031
         BEGIN                                                      02032
         typeas($"
         Format error in last name statement
         ");                                                        02033
         *work* ← SF(stid) SE(stid);                                02034
         typeas($work);                                             02035
         crlf();                                                    02036
         IF NOT contonerr THEN RETURN(FALSE);                       02037
```

```
        END;                                                    02038
      RETURN(TRUE);                                             02039
      END;                                                      02040
   IF (stid ← namelook(idstid, $"'individuals'")) NOT= endfil AND
   (stid := getsub(stid)) NOT= stid THEN                        02041
      LOOP                                                      02042
         BEGIN                                                  02043
         IF FIND SF(stid) $SP '( $SP '' ↑z1 [')] < CH $SP '' > ↑z2
         [')] $SP "LAST NAME" THEN                              02044
            BEGIN                                               02045
            *uplname* ← z1 z2;                                  02046
            IF (stid := getsub(stid)) NOT= stid THEN            02047
               LOOP                                             02048
                  BEGIN                                         02049
                  IF NOT xvinds(stid, $uplname, contonerr) THEN
                  RETURN(FALSE);                                02050
                  IF getftl(stid) THEN                          02051
                     BEGIN                                      02052
                     stid ← getsuc(stid);                       02053
                     EXIT;                                      02054
                     END;                                       02055
                  stid ← getsuc(stid);                          02056
                  END                                           02057
            ELSE                                                02058
               BEGIN                                            02059
               typeas($"
               No substructure for last name statement
               ");                                              02060
               *work* ← SF(stid) SE(stid);                      02061
               typeas($work);                                   02062
               crlf();                                          02063
               IF NOT contonerr THEN RETURN(FALSE);             02064
               END;                                             02065
            END                                                 02066
         ELSE                                                   02067
            BEGIN                                               02068
            typeas($"
            Format error in last name statement
            ");                                                 02069
            *work* ← SF(stid) SE(stid);                         02070
            typeas($work);                                      02071
            crlf();                                             02072
            IF NOT contonerr THEN RETURN(FALSE);                02073
            END;                                                02074
         IF inpstp %↑S% THEN                                    02075
            BEGIN                                               02076
            inpstp ← FALSE;                                     02077
            typeas($"
            Completed Processing for last name ");              02078
            typeas($uplname);                                   02079
            typeas($", ");                                      02080
            *cntsr* ← STRING(cntbleft(stid));                   02081
            typeas($cntsr);                                     02082
            typeas($" last names yet to process.
            ");                                                 02083
            END;                                                02084
```

```
        IF getftl(stid) THEN EXIT;                                      02085
        stid ← getsuc(stid);                                            02086
        END                                                             02087
    ELSE                                                                02088
        BEGIN                                                           02089
        IF stid = endfil THEN                                           02090
            typeas($"
            No 'INDIVIDUALS' branch
            ")                                                          02091
        ELSE                                                            02092
            typeas($"
            'INDIVIDUALS' branch has no substructure
            ");                                                         02093
        IF NOT contonerr THEN RETURN(FALSE);                            02094
        END;                                                            02095
    IF NOT fileno THEN                                                  02096
        close(idstid.stfile := 0);                                      02097
    RETURN(TRUE);                                                       02098
    END.                                                                02099
                                                                        02100
(xvinds) %auxiliary routine for verinds%
PROCEDURE (stid, uplname, contonerr);                                   02101
    LOCAL  i, rstid;                                                    02102
    LOCAL STRING work[1000], idstr[50], idlname[50], uname[50];         02103
    REF uplname;                                                        02104
    *work* ← SF(stid) SE(stid);                                         02105
    getiid($work, $idstr, 0,0);                                         02106
    IF NOT oldid($idstr, stid.stfile) THEN                              02107
        BEGIN                                                           02108
        typeas($"
        OLDID failed for ");                                            02109
        typeas($idstr);                                                 02110
        crlf();                                                         02111
        IF NOT contonerr THEN RETURN(FALSE);                            02112
        END;                                                            02113
    IF NOT ckident($idstr, 0, stid.stfile  : rstid) THEN                02114
        BEGIN                                                           02115
        typeas($"
        CKIDENT failed for ");                                          02116
        typeas($idstr);                                                 02117
        crlf();                                                         02118
        IF NOT contonerr THEN RETURN(FALSE);                            02119
        END;                                                            02120
    IF stid NOT= rstid THEN                                             02121
        BEGIN                                                           02122
        typeas($"
        More than one  ");                                              02123
        typeas($idstr);                                                 02124
        crlf();                                                         02125
        IF NOT contonerr THEN RETURN(FALSE);                            02126
        END;                                                            02127
    getilname($work, $idlname, 0,0);                                    02128
    FOR i ← 1 UP UNTIL > idlname.L DO                                   02129
        IF *idlname*[i] = SP THEN *idlname*[i] ← '';                    02130
    IF *uplname* # *idlname* THEN                                       02131
        BEGIN                                                           02132
```

```
      typeas($"
      Lastname in ident entry not the same as branch name.
      ");                                                          02133
      typeas($"last name in branch head: ");                       02134
      typeas($uplname);                                            02135
      typeas($"
      last name in ident entry: ");                                02136
      typeas($idlname);                                            02137
      typeas($"
      ident entry: ");                                             02138
      typeas($idstr);                                              02139
      crlf();                                                      02140
      IF NOT contonerr THEN RETURN(FALSE);                         02141
      END;                                                         02142
   IF (ldelivery(rstid)).delol THEN %check for valid user name%    02143
      BEGIN                                                        02144
      luser($work, $uname, 0,0);                                   02145
      astruc($uname);                                              02146
      *uname* ← *uname*, 0;  %for tenex%                           02147
      %check user name%                                            02148
         r1 ← 1;                                                   02149
         r2 ← $uname + 1 .V 440700B6;                              02150
         !JSYS stdir;                                              02151
         !JRST ving;                                               02152
         !JRST ving;                                               02153
         !JRST vig;                                                02154
         (ving):  %no good user name%                              02155
            typeas($"
            Invalid username (");                                  02156
            typeas($uname);                                        02157
            typeas($") in ident entry ");                          02158
            typeas($idstr);                                        02159
            crlf();                                                02160
            IF NOT contonerr THEN RETURN(FALSE);                   02161
         (vig): %good user name%                                   02162
      END;                                                         02163
   RETURN(TRUE);                                                   02164
   END.                                                            02165
                                                                   02166
(vergroups) %verify 'groups' branch%
PROCEDURE (fileno, contonerr, identstid, useckident);              02167
   LOCAL idstid, stid;                                             02168
   typeas($"
   Verifying 'GROUPS':
   ");                                                             02169
   idstid ← orgstid;                                               02170
   IF NOT fileno THEN                                              02171
      BEGIN                                                        02172
      ON SIGNAL ELSE close(idstid.stfile := 0);                    02173
      idstid.stfile ← open(0, jflname($"identfile"));              02174
      END                                                          02175
   ELSE                                                            02176
      idstid.stfile ← fileno;                                      02177
   IF (stid ← namelook(idstid, $"'groups'")) NOT= endfil AND (stid :=
   getsub(stid)) NOT= stid THEN                                    02178
      verga(stid, $jgrptst, contonerr, identstid, useckident);     02179
```

```
   IF NOT fileno THEN                                                    02180
      close(idstid.stfile := 0);                                         02181
   RETURN(TRUE);                                                         02182
   END.                                                                  02183
(verorgs) %verify 'organizations' branch%
PROCEDURE (fileno, contonerr, identstid, useckident);                    02184
   LOCAL idstid, stid;                                                   02185
   typeas($"
   Verifying 'ORGANIZATIONS': ");                                        02186
   idstid ← orgstid;                                                     02187
   IF NOT fileno THEN                                                    02188
      BEGIN                                                              02189
      ON SIGNAL ELSE close(idstid.stfile := 0);                          02190
      idstid.stfile ← open(0, jflname($"identfile"));                    02191
      END                                                                02192
   ELSE                                                                  02193
      idstid.stfile ← fileno;                                            02194
   IF (stid ← namelook(idstid, $"'organizations'")) NOT= endfil AND
   (stid := getsub(stid)) NOT= stid THEN                                 02195
      verga(stid, $orgtst, contonerr, identstid, useckident);            02196
   IF NOT fileno THEN                                                    02197
      close(idstid.stfile := 0);                                         02198
   RETURN(TRUE);                                                         02199
   END.                                                                  02200
                                                                         02201
(verga) %support routine for 'groups' or 'organizations' branch%
PROCEDURE (stid, tstproc, contonerr, identstid, useckident);             02202
   %                                                                     02203
   Verify 'group' or 'organizations' branch (or part of plex).  Takes
   stid of starting point (goes to end of plex), address of test
   procedure, and continue-on-error flag.  Performs the following
   tests:                                                                02204
      oldid and ckident                                                  02205
      tstproc                                                            02206
      for each ident in memlist,                                         02207
         oldid and ckident (if useckident)                               02208
   %                                                                     02209
   LOCAL rstid;                                                          02210
   LOCAL STRING cntsr[10], work[1000], idstr[50];                        02211
   REF tstproc;                                                          02212
   IF identstid THEN %process a single ident%                            02213
      BEGIN                                                              02214
      IF getup(identstid) NOT= getup(stid) THEN                          02215
         BEGIN                                                           02216
         typeas($"
         Ident not in appropriate branch
         ");                                                             02217
         IF NOT contonerr THEN RETURN(FALSE);                            02218
         END;                                                            02219
      IF NOT xverga(identstid, &tstproc, contonerr, useckident) THEN
      RETURN(FALSE);                                                     02220
      RETURN(TRUE);                                                      02221
      END;                                                               02222
   LOOP                                                                  02223
      BEGIN                                                              02224
      IF NOT xverga(stid, &tstproc, contonerr, useckident) THEN
```

```
      RETURN(FALSE);                                                    02225
      IF inpstp %↑S% THEN                                               02226
         BEGIN                                                          02227
         inpstp ← FALSE;                                                02228
         typeas($"
         Completed Processing for ");                                   02229
         *work* ← SF(stid) SE(stid);                                    02230
         getiid($work, $idstr, 0,0);                                    02231
         typeas($idstr);                                                02232
         typeas($", ");                                                 02233
         *cntsr* ← STRING(cntbleft(stid));                              02234
         typeas($cntsr);                                                02235
         typeas($" yet to process.
         ");                                                            02236
         END;                                                           02237
      IF getftl(stid) THEN                                              02238
         EXIT;                                                          02239
      stid ← getsuc(stid);                                              02240
      END;                                                              02241
   RETURN(TRUE);                                                        02242
   END.                                                                 02243
                                                                        02244
(xverga) %support routine for verga%
PROCEDURE (stid, tstproc, contonerr, useckident);                       02245
   LOCAL rstid;                                                         02246
   LOCAL TEXT POINTER z1,z2,z3;                                         02247
   LOCAL STRING work[1000], idstr[50], memlist[500],memid[50],
   typstr[50];                                                          02248
   REF tstproc;                                                         02249
   *work* ← SF(stid) SE(stid);                                          02250
   getiid($work, $idstr, 0,0);                                          02251
   IF NOT oldid($idstr, stid.stfile) THEN                               02252
      BEGIN                                                             02253
      typeas($"
      OLDID failed for ");                                              02254
      typeas($idstr);                                                   02255
      crlf();                                                           02256
      IF NOT contonerr THEN RETURN(FALSE);                              02257
      END;                                                              02258
   IF NOT ckident($idstr, 0, stid.stfile  : rstid) THEN                 02259
      BEGIN                                                             02260
      typeas($"
      CKIDENT failed for ");                                            02261
      typeas($idstr);                                                   02262
      crlf();                                                           02263
      IF NOT contonerr THEN RETURN(FALSE);                              02264
      END;                                                              02265
   IF stid NOT= rstid THEN                                              02266
      BEGIN                                                             02267
      typeas($"
      More than one  ");                                                02268
      typeas($idstr);                                                   02269
      crlf();                                                           02270
      IF NOT contonerr THEN RETURN(FALSE);                              02271
      END;                                                              02272
   IF NOT tstproc($work, 0) THEN                                        02273
```

```
      BEGIN                                                          02274
      typeas($"
      Test Procedure failed for ");                                  02275
      typeas($idstr);                                                02276
      crlf();                                                        02277
      IF NOT contonerr THEN RETURN(FALSE);                           02278
      END;                                                           02279
   %check coordinator%                                               02280
      geticord($work, $memid, 0,0);                                  02281
      IF NOT memid.L THEN                                            02282
         BEGIN                                                       02283
         typeas($"
         No cordinator for ");                                       02284
         typeas($idstr);                                             02285
         crlf();                                                     02286
         IF NOT contonerr THEN RETURN(FALSE);                        02287
         END                                                         02288
      ELSE                                                           02289
         IF NOT oldid($memid, stid.stfile) THEN                      02290
            BEGIN                                                    02291
            typeas($"
            OLDID failed for cordinator ");                          02292
            typeas($idstr);                                          02293
            typeas($" : ");                                          02294
            typeas($memid);                                          02295
            crlf();                                                  02296
            IF NOT contonerr THEN RETURN(FALSE);                     02297
            END;                                                     02298
      IF useckident AND NOT ckident($idstr, 0, stid.stfile ) THEN
                                                                     02299
         BEGIN                                                       02300
         typeas($"
         CKIDENT failed for cordinator ");                           02301
         typeas($idstr);                                             02302
         typeas($" : ");                                             02303
         typeas($memid);                                             02304
         crlf();                                                     02305
         IF NOT contonerr THEN RETURN(FALSE);                        02306
         END;                                                        02307
   % Check if type field present if this is an organization. %       02308
      IF &tstproc = $orgtst THEN                                     02309
         BEGIN                                                       02310
         % We are checking orgainizations: check type field. %       02311
         getityp($work, $typstr, 0, 0);                              02312
         IF NOT typstr.L THEN                                        02313
            BEGIN                                                    02314
            typeas($"
            No organization type field for ");                       02315
            typeas($idstr);                                          02316
            crlf();                                                  02317
            IF NOT contonerr THEN RETURN(FALSE);                     02318
            END                                                      02319
         ELSE % Check validity %                                     02320
            BEGIN                                                    02321
            IF NOT( FIND SF(*typstr*) ('I) "ndependent"/('A)
            "ssociate"/('S) "erver"/ ('T) "ip"/ ('U)"ser")) THEN
```

```
                                                                    02322
            BEGIN                                                   02323
            typeas($"
            Organization field invalid for ");                      02324
            typeas($idstr);                                         02325
            typeas($" : ");                                         02326
            typeas($typstr);                                        02327
            crlf();                                                 02328
            IF NOT contonerr THEN RETURN(FALSE);                    02329
            END;                                                    02330
         END;                                                       02331
      END;                                                          02332
getimem($work, $memlist, 0,0);                                      02333
UNTIL memlist.L = empty DO                                          02334
   BEGIN                                                            02335
   IF FIND SF(*memlist*) $SP ↑z1 1$(LD/'-/'') ↑z2 $SP ('( ')/
   $SP/ TRUE) (; $SP/ TRUE) ↑z3 THEN                                02336
      BEGIN                                                         02337
      *memid* ← z1 z2;                                              02338
      *memlist* ← z3 SE(*memlist*);                                 02339
      END                                                           02340
   ELSE                                                             02341
      BEGIN                                                         02342
      typeas($"
      Error in processing membership list for ");                   02343
      typeas($idstr);                                               02344
      typeas($" : ");                                               02345
      typeas($memlist);                                             02346
      crlf();                                                       02347
      IF NOT contonerr THEN RETURN(FALSE);                          02348
      EXIT LOOP;                                                    02349
      END;                                                          02350
   IF NOT oldid($memid, stid.stfile) THEN                           02351
      BEGIN                                                         02352
      typeas($"
      OLDID failed for ");                                          02353
      typeas($idstr);                                               02354
      typeas($" : ");                                               02355
      typeas($memid);                                               02356
      crlf();                                                       02357
      IF NOT contonerr THEN RETURN(FALSE);                          02358
      END;                                                          02359
   IF useckident AND NOT ckident($idstr, 0, stid.stfile ) THEN
                                                                    02360
      BEGIN                                                         02361
      typeas($"
      CKIDENT failed for ");                                        02362
      typeas($idstr);                                               02363
      typeas($" : ");                                               02364
      typeas($memid);                                               02365
      crlf();                                                       02366
      IF NOT contonerr THEN RETURN(FALSE);                          02367
      END;                                                          02368
   END;                                                             02369
RETURN(TRUE);                                                       02370
END.                                                                02371
```

```
   (cntbleft) %count the number of branches left in a plex%
   PROCEDURE (stid);                                                    02372
      LOCAL i;                                                          02373
      i ← 0;                                                            02374
      IF stid.stpsid = origin OR stid = endfil THEN RETURN(0);          02375
      UNTIL getftl(stid) DO                                             02376
         BEGIN                                                          02377
         stid ← getsuc(stid);                                           02378
         BUMP i;                                                        02379
         END;                                                           02380
      RETURN(i);                                                        02381
      END.                                                              02382
FINISH

                                                                        02383
%FORMAT OF IDENTFILE                                                    02384
   The identfile is an NLS file <IDENTFILE>IDENTS.MASTER of the
   following format                                                     02385
      origin statement                                                  02386
         ('usedids')                                                    02387
            number,ident,ident,ident,...,ident,                         02388
               where there are number idents in the statement.  when
               number reaches 200, a new statement is created.          02389
            .
            .
            .                                                           02390
         ('individuals')                                                02391
            ('lastname') LAST NAME                                      02392
               (ident) OrganizationIdent
               Lastname, Firstname MiddleInitial. (nickname), Jr.
               Address

                                                                        02393
               where all of the idents have the same lastname (the
               name of the head of the branch).                         02394
            .
            .
            .                                                           02395
         .
         .
         .                                                              02396
         ('groups')                                                     02397
            (ident) Expand Group
            membershiplist
            groupname
            cordinatorIdent
            Address

                                                                        02398
         .
         .
         .                                                              02399
         ('organizations')                                              02400
            (ident) Organization
            membershiplist
            groupname
```

CMLCO7.12

(MLK)CMLCOMP (MLK)CMLCOMP (MLK)CMLCOMP (MLK)CMLCOMP (MLK)CMLCOMP
(MLK)CMLCOMP (MLK)CMLCOMP (MLK)CMLCOMP (MLK)CMLCOMP (MLK)CMLCOMP

```
< NLS, CMLCOMP.NLS;10, >, 22-OCT-74 10:18 KJM ;;;;
FILE cml CHECK % meta <irby>cml.rel %% (meta,) (irby,cml.rel,) %
   META file
      % **** CHANGE DATE WHEN COMPILING **** (,,018) %
   % DECLARATIONS %
      DUMMY: alter  succ option anyof dumy;
      ERROR: -> "END." $(subsys) "FINIS" :fin[]*;
      SIZE: S=3000 M=100 K=100 N=1000 L=300 G=100;
      SET:
         % RECOGNIZERS %
            KEYOP=1B10          % keyword recognition operator %
            CONFIRM=2B10        % get command confirmation %
            SSEL=3B10  % source selection %
            DSEL=4B10  % destination selection %
            LSEL=5B10  % literal selection %
            CA=6B10             % CA char %
            VIEWSPECS=7B10      % gets viewspecs %
            LEVADJ=10B10        % get level adjust string %
         % OPTIONAL ELEMENTS %
            OPTION=11B10        % optional parameter %
            ANYOF=12B10         % repeated list with failure %
         % CONTROL ELEMENTS %
            PFCALL=21B10        % parsing function %
            EXECUTE=22B10       % transfer to another point in tree %
            CALL=23B10          % subroutine call %
         % FEEDBACK ELEMENTS %
            FBCLEAR=31B10       % clear feedback buffer %
            ECHO=32B10          % echo noise word string %
            RECHO=33B10         % replace last thing echoed %
         % VALUE MANIPULATIONS %
            STORE=41B10         % store value into variable %
            LOAD=42B10                  % load variable to ptr stack %

            ENTER=43B10         % enter constant into stack %
            VALUEOF=44B10;      % valueof built in function %
      FLAGS:
         attr           % attribute values %
         initloc        % ptr to INITIALIZATION statement %
         termloc        % ptr to TERMINATION statement %
         rentryloc      % ptr to RENTRY statement %
         sav            % id value for command %
         value                  % temporary accumulator %
         top            % address of last generated node %
         alt            % address of alternative node %
         suc;           % address of successor node %
      FIELDS: OP=[6:30] CTL=[3:27]
         VAL=[9:18] LH=[18:18] RH=[18:0] ;
      ATTRIBUTES:
         pfunctname             % name of external parsing function %
         functname              % name of external L10 function %
         rulename               % name of a parse rule %
         varname                % temporary variable %
         litstr;                % .ID has literal associated with it %

   % PARSING RULES %
      % file definition %
```

```
    file = "FILE" .ID <"-CML COMPILER 11/30/73">                    055
       <"-FILE "*1> &NAME &DISCARD                                  056
       :/1,/ % output a dummy word %                                057
       $( rule/ dcls )  % can have global rules or declarations %   058
       #(subsys ) % must have at least one subsystem def %          059
       "FINIS" :fin/] *;                                            060
       %---------                                                   061
       (fin):  fin processing and output the symbol table and
       literals %                                                   062
          fin =>                                                    063
             % output a LLO string for all keyword strings which
             are not external %                                     064
                > "LITS"                                            065
                $SYMS(                                              066
                   ?@ litstr *$                                     067
                      (?NOT ?@ extern *$                            068
                         >↑*$                                       069
                         *L$↑LH *L$,                                070
                         *S$                                        071
                      /TRUE)                                        072
                   /TRUE)                                           073
                > "ENDLITS"                                         074
             % put out any literals and the DDT symbol table %      075
                &TABLES;                                            076
 % Rule name definitions %                                          077
    sysname/.ID/ => % define the rule as an external symbol %       078
       @S rulename *1                                               079
       >↑*1 ← top;                                                  080
    rulname/.ID/ => % define the rule as a local symbol %           081
       @S rulename *1                                               082
       >*1 ← top;                                                   083
 % DCL definitions %                                                084
    dcls = ("DCL" / "DECLARE")                                      085
       &attr ← 1 % default is VARIABLE %                            086
       /dclattr/                                                    087
       .#<',> dclitem :dcllist/$/ '; *;                             088
    dclattr = % declaration attributes %                            089
       ("VARIABLE" &attr ← 1                                        090
       / "FUNCTION" &attr ← 2                                       091
       / "PARSEFUNCTION" &attr ← 3                                  092
       / "EXT-KEYWORD" &attr ← 4                                    093
       / "EXTERNAL" &attr ← 5);                                     094
    dclitem = .ID/ .SR /?IF attr # 4 &FAIL/;                        095
    dcllist /$/ =>                                                  096
       ( ?IF attr = 1 % VARIABLE %                                  097
          $(@S varname *$ >*$ &BSS 1, )                             098
       / ?IF attr = 2 % FUNCTION %                                  099
          $(@S functname *$ )                                       0100
       / ?IF attr = 3 % PARSEFUNCTION %                             0101
          $(@S pfunctname *$ )                                      0102
       / ?IF attr = 4 % EXT-KEYWORD %                               0103
          $(@S extern *$ @S litstr *$ )                             0104
       / ?IF attr = 5 % EXTERNAL %                                  0105
          $(@S extern *$ ));                                        0106
 % SUBSYSTEM definition %                                           0107
```

```
    subsys = "SUBSYSTEM" .ID &LABEL                                    0108
       "KEYWORD" .SR :subs2[ 2 ]                                       0109
       &MARK &sav ← 0 &initloc ← 0  &termloc ← 0  &rentryloc ← 0
                                                                       0110
       #(command / rule  )                                             0111
       &UNMARK &top ← sav *  "END.";                                   0112
    subs2 [.ID, .SR] =>                                                0113
       % define the subsystem identifier as an external symbol
       whose value is the address of the subsystem dispatch record
       %                                                               0114
          >↑*1                                                         0115
       % output the subsystem dispatch record %                        0116
          % word 1: RH = ptr to subsystem name string %                0117
             =*S2,                                                     0118
          % word 2: RH = ptr to start of COMMANDS %                    0119
          % word2: LH = validation code %                             0120
             11041731LH top↑RH\1                                       0121
          % word 3: LH = ptr to termination rule/0, RH = ptr to
          initialization rule /0 %                                     0122
             header[ =termloc, =initloc ]                              0123
          % word 4: LH = ptr to rentry rule/0 %                        0124
             header[ =rentryloc, =0 ];                                 0125
% command definitions %                                               0126
    % rules and commands are the same, except that the parse tree
    for a command is linked onto an alternative list for a
    subsystem, and the alternative to a rule is NULL %                 0127
    command =                                                          0128
        ( "COMMAND"                                                    0129
          .ID &LABEL :defcmd[1] &MARK                                  0130
          '= exp &alt←sav  &suc←0  :eval[1] * &sav ← top               0131
          &UNMARK *  ';                                                0132
        / "INITIALIZATION"                                             0133
          :[?IF initloc # 0 &FAIL]                                     0134
          rule &initloc ← top                                          0135
        / "TERMINATION"                                                0136
          :[?IF termloc # 0 &FAIL]                                     0137
          rule &termloc ← top                                          0138
        / "RENTRY"                                                     0139
          :[?IF rentryloc # 0 &FAIL]                                   0140
          rule &rentryloc ← top ) ;                                    0141
    rule =                                                             0142
       .ID &LABEL                                                      0143
       '= exp &alt←0  &suc←0  :eval[1] *                               0144
        :defcmd[1] *  '; ;                                             0145
    defcmd                                                             0146
       [.ID]       =>                                                  0147
          [  ( ?@ varname *1                                           0148
             / ?@ functname *1                                         0149
             / ?@ pfunctname *1 )                                      0150
             <"previously defined name used as a rule" *1 LOC
             LINE> &FAIL]                                              0151
          rulname[*1];                                                 0152
% expression definition %                                             0153
   exp = .#<'/>subexp :alter[$];                                       0154
   subexp =                                                            0155
      .#factor :succ[$];                                               0156
```

```
    factor =                                                        0157
        '( exp ')/                                                  0158
        '[ exp '] :option[1]//                                      0159
        "DUMMY"                                                     0160
           [ +'/ <"DUMMY not last alternative"> &FAIL]              0161
           :dumy[]//                                                0162
         term ;                                                     0163
% output production rules %                                        0164
    % the principal production element is the unparse rule eval.
    It expects a node of 1 element:   an expression node
    (alter/succ/option/terminal).  The global variables "alt" and
    "suc" are used to identify the alternative and successor paths
    respectively. %                                                 0165
    % the entire parse tree for the expression is built, then it
    is processed in reverse order (from "right to left") so that
    no fixups are required to address any alternative or successor
    nodes. %                                                        0166
    %---------%                                                     0167
    eval                                                            0168
       % process a sequence of alternatives %                       0169
        [alter] =>                                                  0170
           ?*Q1 = 1 % single subnode %                              0171
              eval[*1:1]                                            0172
           / % multiple subnodes %                                  0173
              $*1←( % select all alternatives beginning with the
              last, invoking the eval rule recursively to
              process them.%                                        0174
                 ! suc ( eval[*$] )                                 0175
                 &alt ← top                                         0176
                 );                                                 0177
       % process a sequence of successors %                        0178
        [succ] =>                                                   0179
           ?*Q1 = 1 % single subnode %                              0180
              eval[*1:1]                                            0181
           / % multiple subnodes %                                  0182
              $*1←( % select all successors beginning with the
              last, invoking the eval rule recursively to
              process them.%                                        0183
                 ((?LAST  eval[*$])                                 0184
                 / (! alt (&alt ← 0 eval[*$]) ))                    0185
                 &suc ← top                                         0186
                 );                                                 0187
       % generate code for an optional expression %                 0188
        [option[-]]      =>                                         0189
           ! alt, suc ( &alt ← 0 &suc ← 0   eval[*1:1] )            0190
           header[=suc, =suc]                                       0191
           OPTION top↑RH\1                                          0192
           &top ← .-2;                                              0193
       % generate code for a '$ expression %                       0194
        [anyof[-]]       =>                                         0195
           ! alt, suc ( &alt ← 0 &suc ← 0   eval[*1:1] )            0196
           header[=suc, =suc]                                       0197
           ANYOF top↑RH\1                                           0198
           &top ← .-2;                                              0199
       % generate code for a function %                            0200
        [function] => *1;                                           0201
```

```
        % generate code for an assignment %                              0202
           [assign] => *1;                                               0203
        % generate code for a loop construct %                           0204
           [perform] => *1;                                              0205
        % set alternative  for DUMMY node %                              0206
           [dumy] => &top ← suc;                                         0207
        % generate code for any terminal nodes. %                        0208
           [-] =>                                                        0209
              &top ← .                                                   0210
              header[=suc, =alt]                                         0211
              *1;                                                        0212
%---------                                                              0213
(header):  header generates a one word header consisting of two
pointers.  The relocation bits are appropriately set to relocate
either the right or left half words independently %                     0214
   header                                                               0215
      [0,0]       => 0\0;                                               0216
      [0,-]       => 0↑LH  *N2↑RH\1;                                    0217
      [-,0]       => *N1↑LH  0↑RH\2;                                    0218
      [-,-]       => *N1↑LH  *N2↑RH\3;                                  0219
% terminal type  productions %                                          0220
   assign                                                               0221
      [.ID,-] =>                                                        0222
         !alt (&alt ← 0 eval[store[*1]])                                0223
         &suc ← top                                                     0224
         eval[ *2 ]                                                     0225
         &suc ← top;                                                    0226
   call                                                                 0227
      [.ID] =>                                                          0228
         CALL *V1,;                                                     0229
      [.ID,-] =>                                                        0230
         CALL *N2↑VAL *V1,;                                             0231
   echo[.SR] =>                                                         0232
      @S noddt *1                                                       0233
      ECHO =*S1,;                                                       0234
   endcmd[] =>                                                          0235
      CONFIRM,;                                                         0236
   enter                                                                0237
      [-] =>                                                            0238
         ENTER *N1,;                                                    0239
   execute[.ID] =>                                                      0240
      EXECUTE *V1,;                                                     0241
   fbclear[] =>                                                         0242
      FBCLEAR,;                                                         0243
   function                                                             0244
      [.ID,succ] =>                                                     0245
         % check for id used as a rule name or as a variable name
         %                                                              0246
            [ ¤[#@ varname *1/ ?@ rulename *1) <"ERROR: illegal
            use of rule/variable name: " *1 LOO LINE> ]                 0247
         % mark the .ID with the "functname" attribute if not set
         to pfunctname, generate the appropriate type of call  %
                                                                        0248
            (?@ pfunctname *1 eval[pfcall[*1,=*Q2]]                     0249
            / @S functname *1 eval[call[*1,=*Q2]])                      0250
         &suc ← top                                                     0251
```

```
        !alt ( &alt ← 0 eval[*2])                                    0252
        &suc ← top;                                                  0253
     [internal,succ] =>                                              0254
        eval[*1]                                                     0255
        &suc ← top                                                   0256
        !alt ( &alt ← 0 eval[*2])                                    0257
        &suc ← top;                                                  0258
  internal[-] =>                                                     0259
     *N1,;                                                           0260
  keyid[.ID] =>                                                      0261
     KEYOP =*S1,;                                                    0262
  keyw[.SR,-] =>                                                     0263
     @S litstr *1  % mark it as a literal %                          0264
     KEYOP *N2;/512↑CTL *N2↑VAL *V1,;                                0265
  perform % looping construct %                                      0266
     [.ID,-,-] =>                                                    0267
        % check to make sure .ID is a rule %                         0268
           [(?NOT ?@ defned *1 @S  rulename *1                       0269
           /?NOT ?@ rulename *1 <"ERROR: ID must be a rule name"
           *1 LOC LINE> &FAIL)]                                      0270
        % generate code to (1) EXECUTE the rule and (2) evaluate
        the test expression.  Like all CML code, this sequence
        is generated in reverse order so that tree meta can do
        the appropriate linking %                                    0271
           ! alt, suc ( &alt ← 0                                     0272
             eval[ succ[ *2 ] ]                                      0273
             &suc ← top &alt ← . eval[ execute[*1] ] )               0274
           SUC < TOP;                                                0275
  pfcall                                                             0276
     [.ID] =>                                                        0277
        PFCALL *V1,;                                                 0278
     [.ID,-] =>                                                      0279
        PFCALL *N2↑VAL *V1,;                                         0280
  recho[.SR] =>                                                      0281
     @S noddt *1                                                     0282
     RECHO =*S1,;                                                    0283
  store[.ID] =>                                                      0284
      STORE *V1,;                                                    0285
  valueof                                                            0286
     [.SR] =>                                                        0287
        @S litstr *1                                                 0288
        ENTER *V1,;                                                  0289
% terminal nodes for compiler %                                     0290
  term =                                                             0291
     (subname/ confirm/ feedback/ recognition/ loop) ;               0292
  subname =                                                          0293
     .ID                                                             0294
        (('← % assignment statement %                                0295
          % check lhs variable type %                                0296
             (?@ defned                                              0297
                ( ?@ varname                                         0298
                /  <"ERROR: illegal lhs variable: " *1 LOC
                LINE> &FAIL)                                         0299
             / @S varname  @S extern)                                0300
          param  :assign[2] )                                        0301
        / ('( .@<',>param :succ[$] ')                                0302
```

```
            :function[2] ) % function invocation %                    0303
         / :loadid[1]  ) ;                                            0304
         loadid                                                       0305
            [.ID] =>                                                  0306
               (?@ rulename *1                                        0307
                execute[*1])                                          0308
               / (?@ varname *1                                       0309
                load[*1])                                             0310
               / (?NOT ?@ defned *1                                   0311
                @S rulename *1                                        0312
                execute[*1])                                          0313
               / (?@ litstr *1                                        0314
                @S rulename *1                                        0315
                execute[*1])                                          0316
               / <"ERROR: illegal use for identifier:  " *1 LOC
               LINE>;                                                 0317
         load                                                         0318
            [.ID] =>                                                  0319
               (?@ varname *1 LOAD *V1,);                             0320
recognition = keyword/ builtinrec;                                    0321
   keyword = .SR &value ← 3000B % set TNLS & DNLS bits %              0322
      [ '! #qualifier '! ]                                            0323
      :keyw[1, =value];                                               0324
   builtinrec =                                                       0325
      ( ("SSEL" &value ← SSEL/                                        0326
         "DSEL" &value ← DSEL/                                        0327
         "LSEL" &value ← LSEL)                                        0328
         '( param  :function[internal[=value],succ[1]] ') )
                                                                      0329
      /                                                               0330
         "CA" :internal[CA]/                                          0331
         "VIEWSPECS" :internal[VIEWSPECS]/                            0332
         "LEVADJ" :internal[LEVADJ];                                  0333
feedback =                                                            0334
   '<                                                                 0335
      (  "..." .SR '> :recho[1] /                                     0336
         .SR      '> :echo[1] )  /                                    0337
   "CLEAR" :fbclear[];                                                0338
confirm =                                                             0339
   "CONFIRM"                                                          0340
   :endcmd[] ; % call routine to terminate cmd %                      0341
loop =                                                                0342
   "PERFORM"  .ID                                                     0343
      "UNTIL"  &value ← 1                                             0344
      '( exp ') :perform[2,=value];                                   0345
qualifier =                                                           0346
   ("L1" &value ← value+4000B/                                        0347
   "NOTD" &value ← value-2000B % clear DNLS bit %/                    0348
   "NOTT" &value ← value-1000B % clear TNLS bit %/                    0349
   .NUM   VALUE < VALUE**N1 ? DISCARD);                                 0350
param =                                                               0351
   "VALUEOF" '( .SR :valueof[1] ')                                    0352
   / '# .SR :valueof[1]                                               0353
   / "NULL" :enter[=0]                                                0354
   / "TRUE" :enter[=1]                                                0355
   / "FALSE" :enter[=0]                                               0356
```

```
          / factor;                                                    0357
    END of CML                                                         0358
% runfil to create a new cml compiler %                               0359
    DEL CML.SAV;*<ESC>                                                 0360
                                                                       0361
    EXP                                                                0362
                                                                       0363
    TENLDR                                                             0364
    /S                                                                 0365
    /M                                                                 0366
    /4000100                                                           0367
    <META>LIBE                                                         0368
    CML                                                                0369
    <ESC>                                                              0370
    DDT                                                                0371
    MOVE 1,116<ESC>X                                                   0372
    MOVEM 1,400000<ESC>X                                               0373
    INITLL<ESC>G                                                       0374
    MESSAGE                                                            0375
    JUNK                                                               0376
                                                                       0377
                                                                       0378
    SS 400 440 CML.SAV;<ESC>                                           0379
                                                                       0380
    DEL JUNK.REL;1<ESC>                                                0381
                                                                       0382
    DEL XXX.REL;*<ESC>                                                 0383
                                                                       0384
    EXP                                                                0385
                                                                       0386
```

CSENDMAIL

(MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)
(MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)CSENDMAIL

```
< NLS, CSENDMAIL.NLS;27, >, 4-OCT-74 13:21 CHI ;;;;( NLS,
CSENDMAIL.NLS;23, ), 10-JUN-74 13:23 CHI ;
FILE csendmail % 110 to <rel-nls>csendmail % % (110.sav,)  (rel-nls,
csendmail.rel,) %                                                    02
%....Declarations....%                                               03
   REGISTER r1=1, r2=2, r3=3, r4=4, r5=5, r6=6, r7=7;                04
   DECLARE    %for Journal invocation from another (e.g. FTP) process%
                                                                     05
      ok=1,          % successful return %                          06
      cat=0,         % catastrophic error return %                  07
      submission=1, % journal submission %                         08
      leaving=-1,    % file leaving system %                        09
      entering=0;    % file entering system %                      010
   DECLARE STRING nullsource = "/No content was specified!/";      01253
%....Initialize JWORK file....%                                     011
   (initjwork) %Initialize  JWORK file%
   PROCEDURE;                                                       012
      % DOCUMENTATION for initjwork:                               013
         This procedure initializes the JWORK file, which is the work
         file for sending or forwarding Jounal items.              014
            The JWORK file is named a concatenation of
            "/send-mail/.IDENT , where IDENT is the ident of the person
            using NLS.                                             015
            The JWORK file is created in the logged in directory by
            xjloaworfil.  It is assumed that the file is NULL (has only
            simple origin statement) at the time this routine is
            called.                                                016
         This routine initializes the origin statement as the Journal
         header.                                                   017
            Replace existing origin statement with:                018
               Journal Number                                      019
                  (a dummy meaning deferred number assingment)     020
               Author                                              021
                  (ident of user)                                  022
               Message flag                                        023
                  (this will be changed when user specifies source of
                  item being sent or forwarded)                    024
               Clerk                                               025
                  (ident of user)                                  026
               Journal Directives                                  027
               2 EOL's (for delimiting comments later)             028
         Finally, it inserts a dummy message as statement 1.       029
         %                                                         030
      LOCAL stid;                                                  031
      %--------------------%                                       032
      %check jworkstid and setup jwp1%                             033
         jworkstid.stpsid ← origin;  %just for safety's sake%      034
         IF NOT goodrng(jworkstid) THEN                            035
            err($"Journal Work File Cannot be Initialized");       036
         FIND SF(jworkstid) ↑jwp1;                                 037
      %make sure name delimiters are right%                      01499
         csetnsta(jworkstid, '(, ') );                           01500
      %set up Journal Header%                                      038
         ST jwp1← ";;;; F (J) ; Author(s): /", *initsr*, ';,       039
            " Clerk: ", *initsr*, ';,                              040
            %Journal Formatting Directives followed by two EOL's%  041
```

```
            "  .IGD=0; .SNF=HJRM; .RM=HJRM-7;  .PN=-1; .YBS=1;
            .PES;", EOL, EOL;                                          042
      %insert dummy message%                                           043
         IF (stid ← getsub(jworkstid)) NOT= jworkstid THEN             044
            BEGIN %there is substructure, delete it%                   045
            cdelgro(stid, getail(stid), FALSE, 0);                     046
            END;                                                       047
         cis(jworkstid, $nullsource, $"d");                            048
      RETURN END.
                                                                       049
%....Set contents of Journal header fields....%                        050
   % DOCUMENTATION FOR setj routines:                                  051
      These routines are used to set values of fields in the Journal
      header statement in JWORK file.                                  052
      They assume JWP1 is set up to point to the first character of the
      origin statement of the JWORK file and are, in general, called
      with a single parameter, the address of a STRING which contains
      the value to which the field is to be set.                       053
      They subsequently search the header statement for the field (or
      the proper location of the field if the field is optional and
      non-present), and replace the existing field with the string (or
      create the field if necessary).                                  054
      Where there are a number of optional fields immediately adjacent
      to each other, the search algorithm uses a list of alternatives,
      searching for the first preceding field first, and if that fails,
      the second preceding field, and if that fails.....               055
      Setj routines which do unusual things will be separately
      documented.                                                      056
      %                                                                057
   (setjauthor) %set the author field of JWORK file%
   PROCEDURE (authorstr);                                              058
      LOCAL TEXT POINTER z1, z2;                                       059
      REF authorstr;  %address of string containing ident(s) of new
      author(s)%                                                       060
      %--------------------%                                          061
      astruc(&authorstr); %Set it upper case%                          062
      FIND jwp1 > ["Author(s):"] ['/] ↑z1 [';] ↑z2 ←z2;                063
      ST z1 z2 ← *authorstr*;                                          064
      RETURN;                                                          065
      END.
                                                                       066
   (setjaction) %set Action Distribution field in JWORK header%
   PROCEDURE (recipientstr);                                           067
      REF recipientstr;  %address o string containling list of,"action"
      recipients%                                                      068
      LOCAL TEXT POINTER z1,z2;                                        069
      %--------------------%                                          070
      FIND jwp1 >                                                      071
        . (["Action Distribution: "] < ['A] SP ↑z1 > [';] ↑z2          072
          / ["Author(s): "] [';] ↑z1 ↑z2);                             073
      ST z1 z2 ←  " Action Distribution: /", *recipientstr*, ';;       074
      RETURN END.
                                                                       075
   (setjaxcess) %set the access list field in JWORK header%
   PROCEDURE (accessstr);                                              076
      %----------%                                                    077
```

```
   LOCAL TEXT POINTER z1, z2;                                          078
   REF accessstr;                                                      079
   %----------%                                                        080
   FIND jwp1 > (["AccessList: "] < ['A] CH ↑z1 > [';] ↑z2 /            081
      (["Updates Document(s): "] /                                     082
      ["Obsoletes Document(s): "] /                                    083
      ["RFC# "] /                                                      084
      ["Sub-Collections: "] /                                          085
      ["KeyWords: "] /                                                 086
      ["Distribution: "] /                                             087
      ["Author(s): "]) [';] ↑z1 ↑z2);                                  088
   ST z1 z2 ← " AccessList: ", *accessstr*, ';;                        089
   RETURN;                                                             090
   END.                                                                091
                                                                       092
(setiinfo) %set Information Distribution field in JWORK header%
PROCEDURE (recipientstr);                                              093
   REF recipientstr;  %address o string containling list of
   "information only" recipients%                                      094
   LOCAL TEXT POINTER z1,z2;                                           095
   %--------------------%                                             096
   FIND jwp1 >                                                         097
      (["Information-only Distribution: "] < ['I] SP ↑z1 > [';] ↑z2
                                                                       098
      / ["Author(s): "] [';] ↑z1 ↑z2);                                 099
   ST z1 z2 ←  " Information-only Distribution: /", *recipientstr*,
   ';;                                                                 0100
   RETURN END.
                                                                       0101
(setjcomment) %set Comment field in JWORK header%
PROCEDURE (commentstr);                                                0102
   REF commentstr; %address of string containing desire comment%       0103
   LOCAL TEXT POINTER z1,z2;                                           0104
   %---------------%                                                  0105
   %remove " from the string%                                          0106
      FIND SF(*commentstr*);                                           0107
      LOOP CASE READC OF                                               0108
         = '"': %replace it with a single quote%                       0109
            BEGIN                                                      0110
            FIND ↑z2 < CH > ↑z1;                                       0111
            ST z1 z2 ← '';                                             0112
            FIND z2 >;                                                 0113
            END;                                                       0114
         = ENDCHR: EXIT LOOP;                                          0115
         ENDCASE ;                                                     0116
   FIND jwp1 > ["Clerk: "] ("
                                                                       0117
   "] ↑z1;
   ST jwp1 ← SF(jwp1) z1, ".PEL;  .PN=PN-1; .GCR;", *commentstr*;
                                                                       0118
   RETURN END.
                                                                       0119
(setjexpedite) %set or clear expedite attribute in JWORK header%
PROCEDURE (onoff);                                                     0120
   %IF onoff = TRUE, set expedite attribute, otherwise clear it%       0121
   LOCAL TEXT POINTER z1,z2;                                           0122
```

```
   %---------------%                                                    0123
   FIND jwp1 >                                                        0124
      (["(Expedite) "] ↑z2 < ['(] SP > ↑z1 /                          0125
      (["Title: "] < ['T] / ["Author(s): "] < ['A/) SP > ↑z1 ↑z2);
                                                                      0126
   IF onoff THEN ST z1 z2← " (Expedite) "                             0127
   ELSE ST z1 z2 ← NULL;                                              0128
   RETURN END.
                                                                      0129
(set.number) %set the contents of the Number field in the JWORK
header%
PROCEDURE (numberstr);                                                0130
   REF numberstr;                                                     0131
   LOCAL TEXT POINTER z1,z2;                                          0132
   %---------------%                                                  0133
   FIND jwp1 > [";;;;"] ['(] ('J/'j) ↑z1 [')] ↑z2 ←z2;                0134
   ST z1 z2 ← *numberstr*;                                            0135
   RETURN;                                                            0136
   END.
                                                                      0137
(set.keyword) %set keywords field in JWORK header%
PROCEDURE (keywordstr);                                               0138
   REF keywordstr; %address of string containing list of keywords%
                                                                      0139
   LOCAL TEXT POINTER z1,z2;                                          0140
   %---------------%                                                  0141
   FIND jwp1 >                                                        0142
      (["Keywords: "] < ['K] SP ↑z1 > [';] ↑z2 /                      0143
      (["Distribution: "]/["Author(s): "]) [';] ↑z1 ↑z2);              0144
   ST z1 z2 ← " Keywords: ", *keywordstr*, ';;                        0145
   RETURN END.
                                                                      0146
(set.link) %set INSERT LINK (***FIX THIS***) field in JWORK header%
PROCEDURE (keywordstr);                                               0147
   REF keywordstr; %address of string containing list of keywords%
                                                                      0148
   LOCAL TEXT POINTER z1,z2;                                          0149
   %---------------%                                                  0150
   FIND jwp1 >                                                        0151
      (["Keywords: "] < ['K] SP ↑z1 > [';] ↑z2 /                      0152
      (["Distribution: "]/["Author(s): "]) [';] ↑z1 ↑z2);              0153
   ST z1 z2 ← " Keywords: ", *keywordstr*, ';;                        0154
   RETURN END.
                                                                      0155
(set.rfc) %set RFC field in JWORK header%
PROCEDURE (rfcstr);                                                   0156
   REF rfcstr; %address of string containing RFC number%             0157
   LOCAL TEXT POINTER z1,z2;                                          0158
   %---------------%                                                  0159
   FIND jwp1 >                                                        0160
      (["RFC# "] < ['R] SP ↑z1 > [';] ↑z2 /                           0161
      (["Sub-Collections: "] / ["Keywords: "] / ["Distribution: "] /
      ["Author(s): "]) [';] ↑z1 ↑z2);                                 0162
   ST z1 z2 ← " RFC# ", *rfcstr*, ';;                                 0163
   RETURN END.
                                                                      0164
```

```
(setjsource) %set source type and content in JWORK header%
PROCEDURE (sourcetype, s1, s2);                                          0165
   % souce may be of type statement, group, file, or hardcopy %          0166
   LOCAL TEXT POINTER w1, w2, t1, t2;                                    0167
   LOCAL STRING locstr[200];                                             0168
   REF s1, s2;                                                           0169
   %-----------------%                                                   0170
   w1[1] ← w2[1] ← 1;                                                    0171
   FIND jwp1 > ["";;;; "] ↑t1 CH ↑t2;                                    0172
   FIND t1;                                                              0173
   CASE READC OF                                                         0174
      = 'H:  % remove hardcopy location field %                          0175
         BEGIN                                                           0176
         IF FIND jwp1 > [')] ["Hard Copy--Location: "] < ['H] SP ↑w1
         > [';] ↑w2 THEN                                                 0177
            ST w1 w2 ← NULL;                                             0178
         END;                                                            0179
      = 'F:  % remove origin field %                                     0180
         BEGIN                                                           0181
         IF FIND jwp1 > ["Clerk: "] ["Origin: "] < ['0] SP ↑w1 >
         ["####;"] ↑w2 THEN                                              0182
            ST w1 w2 ← NULL;                                             0183
         END;                                                            0184
      = 'S: NULL;  % just replace substructure is  OK? %                 0185
      ENDCASE err($"Illegal Journal Workfile format -- setjsource");
                                                                         0186
   IF ( w1 ← getsub(jwp1) ) NOT= jwp1 THEN                               0187
      BEGIN  % delete old source %                                       0188
      w2 ← getail(w1);                                                   0189
      cdelgro(w1, w2, FALSE, 0);                                         0190
      END;                                                               0191
   CASE sourcetype OF                                                    0192
      =stmtv:  %statement or text%                                       0193
         BEGIN                                                           0194
          % insert new source %                                          0195
            cinssta(jwp1, levdown, &s1, &s2);                            0196
         ST t1 t2 ← 'F;                                                  0197
         END;                                                            0198
      =groupv:  % group (branch, plex) %                                 0199
         BEGIN % insert new group as substructure of header %            0200
         IF s1.stastr THEN %user typed 'group' text%                     0201
            cinssta(jwp1, levdown, &s1, &s2)                             0202
         ELSE % normal group %                                           0203
            ccopgro(jwp1, levdown, s1, s2, FALSE, 0);                    0204
         ST t1 t2 ← 'F;                                                  0205
         END;                                                            0206
      =filev: % whole file %                                             0207
         BEGIN                                                           0208
         %load the file%                                                 0209
            caddexp(&s1, &s2, lda(), $w1);                               0210
            IF w1 = endfil THEN                                          0211
               BEGIN                                                     0212
               *locstr* ← "Illegal link: ", s1 s2;                       0213
               err($locstr);                                             0214
               END;                                                      0215
            s1 ← orgstid;  s1.stfile ← w1.stfile;                        0216
```

```
        %copy origin statement into jwork header%                    0217
           IF NOT FIND jwp1 ["Clerk:"] [EOL EOL] ↑w1 ←w1 ←w1 THEN
                                                                     0218
              err($"Improper Journal Workfile Format --
              setjsource");                                          0219
           ST w1 w1 ← " Origin: ", SF(s1) SE(s1), "####;";           0220
        %copy substructure to jwork file%                            0221
           IF (s1 := getsub(s1)) # s1 THEN % there is substructure
           in user's file %                                          0222
              BEGIN                                                  0223
              s2 ← getail(s1);                                       0224
              REPEAT CASE (groupv);                                  0225
              END;                                                   0226
           % no substructure in user's file -- boy is he stupid! %
                                                                     0227
              *locstr* ← "[no content was specified!]";              0228
              FIND SF(*locstr*) ↑s1 SE(*locstr*) ↑s2;                0229
              REPEAT CASE (stmtv);                                   0230
        END;                                                         0231
     =forwdv:  % forward an item to specified recipients %           0232
        BEGIN                                                        0233
         % insert forward request %                                  0234
           cinssta(jwp1, levdown, $s1, $s2);                         0235
        ST t1 t2 ← 'S;                                               0236
        END;                                                         0237
     =hcopyv:  % hardcopy item %                                     0238
        BEGIN                                                        0239
        FIND t2 [')] ↑w1;                                            0240
        ST t1 ← SF(t1) t1, 'H, t2 w1, " Hard Copy--Location: ", s1
        s2, ';, w1 SE(w1);                                           0241
        END;                                                         0242
     ENDCASE err(@"Illegal source type in setjsource");              0243
  RETURN;                                                            0244
  END.                                                               0245
(setjsubcol) %set Sub-Collections field in JWORK header%
PROCEDURE (subcollstr);                                              0246
  LOCAL STRING tempsr[150];                                          0247
  REF subcollstr; %address of string containing list of
  Sub-Collections%                                                   0248
  LOCAL TEXT POINTER z1,z2;                                          0249
  %---------------%                                                 0250
  *tempsr* ← *subcollstr*;                                           0251
  astruc($tempsr); %Make sure that it is upper case%                 0252
  FIND SF(jwp1)                                                      0253
     (["Sub-Collections: "] < ['S] SP ↑z1 > [';] ↑z2 /               0254
     (  (\Keywords: "] / ["Distribution: "] / ["Author(s): "]) [';]
     ↑z1 ↑z2);                                                       0255
  ST z1 z2 ← " Sub-Collections: ", *tempsr*, ';;                     0256
  RETURN END.
                                                                     0257
(setjobsoletes) %set Obsoletes Document(s) field in JWORK header%
PROCEDURE (obsoletestr);                                             0258
  LOCAL TEXT POINTER z1, z2;                                         0259
  REF obsoletestr; %address of string containing list of Obsoleted
  Documents%                                                         0260
  %---------------%                                                 0261
```

```
   FIND jwpl >                                                          0262
      (["Obsoletes Document(s): "] < ['O] SP ↑z1 > [';] ↑z2/            0263
      (["RFC# "] / ["Sub-Collections: "] / ["Keywords: "] /
      ["Distribution: "] / ["Author(s): "]) [';] ↑z1 ↑z2);               0264
   ST z1 z2 ← " Obsoletes Document(s): ", *obsoletestr*, '; ;           0265
   RETURN;                                                              0266
   END.
                                                                        0267
(set,unrecorded) %set or clear UNRECORDED attribute in JWORK heade %
PROCEDURE (onoff);                                                      0268
   %IF onoff = TRUE, set unrecorded attribute, otherwise clear it%
                                                                        0269
   LOCAL TEXT POINTER z1,z2;                                            0270
   %---------------%                                                   0271
   FIND jwpl >                                                          0272
      ["(Unrecorded) "] ↑z2 < ['(] SP > ↑z1 /                           0273
      (["Title: "] < ['T] / ["Author(s): "] < ['A]) SP > ↑z1 ↑z2);
                                                                        0274
   IF onoff THEN ST z1 z2← " (Unrecorded) "                             0275
   ELSE ST z1 z2 ← NULL;                                                0276
   RETURN END.
                                                                        0277
(set,updates) %set Updates Document(s) field in JWORK header%
PROCEDURE (updatestr);                                                  0278
   LOCAL TEXT POINTER z1, z2;                                           0279
   REF updatestr; %address of string containing list of Updated
   Documents%                                                           0280
   %---------------%                                                   0281
   FIND jwpl >                                                          0282
      (["Updates Document(s): "] < ['U] SP ↑z1 > [';] ↑z2/              0283
      (["Obsoletes Document(s): "] / ["RFC# "] / ["Sub-Collections:
      "] / ["Keywords: "] / ["Distribution: "] / ["Author(s): "])
      [';] ↑z1 ↑z2);                                                    0284
   ST z1 z2 ← " Updates Document(s): ", *updatestr*, '; ;               0285
   RETURN;                                                              0286
   END.
                                                                        0287
(set,title)%set Title field in JWORK header%
PROCEDURE(titlestr);                                                    0288
   REF titlestr; %address of string containing desired title%           0289
   LOCAL TEXT POINTER z1,z2;                                            0290
   %---------------%                                                   0291
   %remove " from the string%                                           0292
      FIND SF(*titlestr*);                                              0293
      LOOP CASE READC OF                                                0294
         = '": %replace it with a single quote%                         0295
            BEGIN                                                       0296
            FIND ↑z2 < CH > ↑z1;                                        0297
            ST z1 z2 ← '';                                              0298
            FIND z2 >;                                                  0299
            END;                                                        0300
         = ENDCHR: EXIT LOOP;                                           0301
         ENDCASE ;                                                      0302
   FIND jwpl >                                                          0303
      (["Title: ,H1="]  < ['T] SP ↑z1 > [';] ↑z2/                       0304
       ["Author(s): "] < ['A] SP ↑z1 ↑z2>);                             0305
```

```
    ST z1 z2 ← " Title: .Hl=", '"', *titlestr*, '"', ';;                    0306
    RETURN END.
                                                                            0307
(setcacc) %set access field in catalog entry%
PROCEDURE (entrystid, repstr, ptr1, ptr2);                                  0308
    %----------%                                                           0309
    LOCAL repcnt;                                                           0310
    LOCAL TEXT POINTER z1, z2;                                              0311
    REF repstr, ptr1, ptr2;                                                 0312
    %----------%                                                           0313
    IF NOT &ptr1 THEN getcacc (entrystid, 0, $z1, $z2)                      0314
    ELSE FIND ptr1 ↑z1 ptr2 ↑z2;                                            0315
    IF *repstr* = *nullfield* THEN %delete the field%                       0316
        IF (FIND z1 < [$PT] $NP > ↑z1 ["AccessList:"] z2 [';] ↑z2)
        THEN                                                                0317
            ST entrystid ← SF(entrystid) z1, z2 SE(entrystid)               0318
        ELSE NULL                                                           0319
    ELSE                                                                    0320
        BEGIN                                                               0321
        repcnt ← repstr.L;                                                  0322
        IF *repstr* [repstr.L] = '; THEN BUMP DOWN repcnt;                  0323
        IF NOT (FIND z1 < [$PT] $NP > ["AccessList:"]) THEN                 0324
            ST entrystid ← SF(entrystid) z1, " AccessList: ", *repstr*
            [1 TO repcnt], ';, z2 SE(entrystid)                             0325
        ELSE                                                                0326
            ST entrystid ← SF(entrystid) z1, *repstr* [1 TO repcnt], z2
            SE(entrystid);                                                  0327
        END;                                                                0328
    RETURN;                                                                 0329
    END.                                                                    0330
                                                                            0331
%....Status....%                                                            0332
    (jstatus) %generate a submission status string%
    PROCEDURE (status);                                                     0333
        % jstatus documentation                                            0334
            It works by calling a set of getj routines, which are roughly
            complementary to the setj routines including the fact that
            they expect jwpl to be set to the first character of the JWORK
            origin statement.                                               0335
            Each getj routine either null's the passed string if the field
            does not exist or fills it with the value of the field.         0336
            The results are simply appended and separated by EOL's to form
            a longer string which is printed out to the user.               0337
            %                                                               0338
        LOCAL stid;                                                         0339
        LOCAL TEXT POINTER z1, z2;                                          01260
        LOCAL STRING str[1500];                                             0340
        REF status;  %address of string into which the status message is
        to be stored%                                                       0341
        %--------------------%                                             0342
        *status* ←                                                          0343
            *[addeol ( getjnumber ( $str ), $sjnumber )]*,                  0344
            *[addeol ( getjaxcess ( $str ), $sjaccess )]*,                  0345
            *[addeol ( getjtitle ( $str ), $sjtitle )]*,                    0346
            *[addeol ( getjcomment ( $str ), $sjcomment )]*,                0347
            *[addeol ( getjauthor ( $str ), $sjauthor )]*,                  0348
```

```
    */addeol ( getjaction ( $str ), $sjaction )]*,                          0349
    */addeol ( getjinfo ( $str ), $sjinfo )]*,                              0350
    */addeol ( getjsubcol ( $str,0 ), $sjsubcol  )]*;                       0351
*status* ← *status*, %this is broken to avoid an L10 STACK
OVERFLOW%                                                                   0352
    */addeol ( getjkeyw ( $str ), $sjkeywords )]*,                          0353
    */addeol ( getjexpedite ($str), $sjhandling)]*,                         0354
    */addeol ( getjunrecorded ($str), $sjrecording)]*,                      0355
    */addeol ( getjrfc ( $str ), $sjrfc )]*,                                0357
    */addeol ( getjobsoletes ( $str ), $sjobsoletes )]*,                    0358
    */addeol ( getjupdates ( $str ), $sjupdates )]*,                        0359
    */addeol ( getjlink ( $str ), $sjlink )]*;                              0360
getjtype( $str );                                                           0361
CASE *str*[1] OF  %type of source%                                          0362
    = 'M, = 'F:                                                            01282
      BEGIN  %determine what status to give%                               01257
      % Is it a file?%                                                     01261
      IF FIND SF(jwpl) ["Clerk: "] ["Origin: "] ↑z1 [';][';] ↑z2           01259
      THEN
         *status* ← *status*, EOL, *sjfile*, z1 z2, EOL                    01262
      ELSE  %see if it is a structure%                                     01263
         IF (stid ← getnxt(jwpl)) NOT= endfil THEN % There is
         substructure%                                                     01254
            IF getnxt(stid) = endfil THEN                                  01255
               IF NOT FIND SF(stid) *nullsource* ENDCHR THEN               01264
                  *status* ← *status*, EOL, *sjmessage*, SF(stid)
                  SE(stid), EOL                                             0363
               ELSE NULL  %user has not supplied source yet%               01265
            ELSE %show some text from first and last branches%
                                                                           01256
               BEGIN                                                       01271
               FIND SF(stid) ↑z1 $15CH ↑z2;                                01272
               IF getftl(stid) THEN % a branch%                            01268
                  *status* ← *status*, EOL, "BRANCH BEGINING: ",
                  '", z1 z2, '", EOL                                       01270
               ELSE                                                        01269
                  BEGIN                                                    01275
                  *status* ← *status*, EOL, "GROUP BEGINING: ",
                  '", z1 z2, '", EOL;                                      01274
                  stid ← getail(stid);                                     01278
                  FIND SF(stid) ↑z1 $15CH ↑z2;                             01276
                  *status* ← *status*, "  [THROUGH]  ", '", z1
                  z2, '", EOL;                                             01279
                  END;                                                     01277
               END;                                                        01273
      END;                                                                 01258
    = 'H:  %hardcopy%                                                      01284
      *status* ← *status*,                                                 01286
         */addeol ( getjhcloc ( $str ), $sjhardcopy )]*;                    0356
    = 'S: %secondary distribution%                                         01285
      IF (stid ← getnxt(jwpl)) NOT= endfil THEN % There is
      substructure%                                                        01288
         *status* ← *status*, EOL, *sjforward*, SF(stid)
         SE(stid), EOL;                                                    01287
    ENDCASE;                                                               01283
RETURN (&status);                                                           0364
```

```
    END.                                                                0365
(addeol) %support routine for jstatus%                                  0366
PROCEDURE (string, heading);                                            0367
    %*addeol*
        This routine is used by jstatus for inserting EOL's between
        fields.                                                         0368
        Basically, if it is handed a NULL string, then it does
        nothing, but if handed a non-empty string, it appends an EOL.
                                                                        0369
        %                                                               0370
    REF string, heading;                                                0371
    %-----------------%                                                 0372
    IF string.L > 0 THEN *string* ← *heading*, SP, *string*, EOL;       0373
    RETURN(&string);                                                    0374
    END.                                                                0375
%....Get contents of Journal header fields....%                         0376
    %                                                                   0377
This is a set of routines which return values of fields in the
Journal header .                                                        0378
Generally, they assume jwp1 is a pointer to the first character of
the statement containing the header, and search that statement for
the presence of the field.                                              0379
If the field is not found, the string address is set to  NULL.          0380
Otherwise, the contents of the field is copied into the passed
string, and its address is returned.                                    0381
In cases where the getj routines do unusual things, they are
documented separately                                                   0382
    %                                                                   0383
(getjaction) %get the contents of the Action Distribution field from
the JWORK header%
PROCEDURE (string);                                                     0384
    REF string;  LOCAL TEXT POINTER z1, z2;                             0385
    IF FIND jwp1 > /"Action Distribution: "] ['/] ↑z1 [';] ↑z2 ←z2
    THEN                                                                0386
        *string* ← z1 z2                                                0387
    ELSE *string* ← NULL;                                               0388
    RETURN(&string);                                                    0389
    END.                                                                0390
(getjauthor) %get the contents of the Author field from the JWORK
header%
PROCEDURE (string);                                                     0391
    REF string;  LOCAL TEXT POINTER z1, z2;                             0392
    FIND jwp1 > /"Author(s):"] ['/] ↑z1 [';] ↑z2 ←z2;                   0393
    *string* ← z1 z2;                                                   0394
    RETURN(&string);                                                    0395
    END.                                                                0396
(getjaxcess) %get the contents of the Access List field from the
JWORK header%
PROCEDURE (string);                                                     0397
    REF string;  LOCAL TEXT POINTER z1, z2;                             0398
    IF rdprvsts (jwp1.stfile) = $psprivate THEN                         0399
        *string* ← "PRIVATE"                                            0400
```

```
   ELSE *string* ← NULL;                                                0401
   RETURN(&string);                                                     0402
   END.

                                                                        0403
(getjclerk) %get the contents of the Clerk field from the JWORK
header%
PROCEDURE (string);                                                     0404
   REF string;  LOCAL TEXT POINTER z1, z2;                              0405
   FIND jwp1 > /"Clerk: "/ ↑z1 /'./ ↑z2 ←z2;                            0406
   *string* ← z1 z2;                                                    0407
   RETURN(&string);                                                     0408
   END.
                                                                        0409
(getjcomment) %get the contents of the Comment field from the JWORK
header%
PROCEDURE (string);                                                     0410
   REF string;  LOCAL TEXT POINTER z1, z2;                              0411
   *string* ← NULL;                                                     0412
   IF ( FIND jwp1 > /"

   "/ /".GCR;"/  ↑z1 ) AND ( POS z1 # SE(z1) ) THEN                     0413
      *string* ← z1 SE(z1);                                             0414
   RETURN(&string);                                                     0415
   END.
                                                                        0416
(getjexpedite) %get the Expedite attribute from the JWORK header%
PROCEDURE (string);                                                     0417
   REF string;  LOCAL TEXT POINTER z1, z2;                              0418
   *string* ← NULL;                                                     0419
   IF FIND jwp1 > /"(Expedite)"/ THEN *string* ← "Expedite";            0420
   RETURN(&string);                                                     0421
   END.
                                                                        0422
(getjunrecorded) %get the Unrecorded attribute from the JWORK
header%
PROCEDURE (string);                                                     0423
   REF string;  LOCAL TEXT POINTER z1, z2;                              0424
   *string* ← NULL;                                                     0425
   IF FIND jwp1 > /"(Unrecorded)"/ THEN *string* ← "Unrecorded";        0426
   RETURN(&string);                                                     0427
   END.
                                                                        0428
(getjhcloc) %get the contents of the Hard Copy Location field from
the JWORK header%
PROCEDURE (string);                                                     0429
   REF string;  LOCAL TEXT POINTER z1, z2;                              0430
   *string* ← NULL;                                                     0431
   IF FIND jwp1 > /"Hard Copy--Location: "/ ↑z1 /';/ ↑z2 ←z2 THEN
                                                                        0432
      *string* ← z1 z2;                                                 0433
   RETURN(&string);                                                     0434
   END.
                                                                        0435
(getjinfo) %get the contents of the Information-only Distribution
field from the JWORK header%
```

```
PROCEDURE (string);                                                    0436
   REF string;  LOCAL TEXT POINTER z1, z2;                             0437
   IF FIND jwp1 > /"Information-only Distribution: "/ ('//) ↑z1 (';/)
   ↑z2 ←z2 THEN                                                        0438
      *string* ← z1 z2                                                 0439
   ELSE *string* ← NULL;                                               0440
   RETURN(&string);                                                    0441
   END.
                                                                       0442
(get.jlink) %get the contents of the Insert Link field from the JWORK
header%
PROCEDURE (string);                                                    0443
   REF string;  LOCAL TEXT POINTER z1, z2;                             0444
   IF FIND jwp1 > /"Insert Link at: "/ ↑z1 (';/) ↑z2 ←z2 THEN          0445
      *string* ← z1 z2                                                 0446
   ELSE *string* ← NULL;                                               0447
   RETURN(&string);                                                    0448
   END.
                                                                       0449
(get.jkeyw) %get the contents of the Keywords field from the JWORK
header%
PROCEDURE (string);                                                    0450
   REF string;  LOCAL TEXT POINTER z1, z2;                             0451
   IF FIND jwp1 > /"Keywords: "/ ↑z1 (';/) ↑z2 ←z2 THEN                0452
      *string* ← z1 z2                                                 0453
   ELSE *string* ← NULL;                                               0454
   RETURN(&string);                                                    0455
   END.

                                                                       0456
(get.jnumber) %get the contents of the Number field from the JWORK
header%
PROCEDURE (string);                                                    0457
   REF string;  LOCAL TEXT POINTER z1, z2;                             0458
   IF NOT FIND jwp1 > /";;;;"//'(/ ('J/'J) ↑z1 $D ↑z2 THEN             0459
      err($"Bad Journal Work File");                                   0460
   *string* ← z1 z2;                                                   0461
   RETURN(&string);                                                    0462
   END.
                                                                       0463
(get.jtype) %get the TYPE of the item from the JWORK header%
PROCEDURE (string);                                                    0464
   REF string;  LOCAL TEXT POINTER z1, z2;                             0465
   IF NOT FIND jwp1 > /";;;; "/ ↑z1 CR ↑z2 THEN                        0466
      err($"Bad Journal Work File");                                   0467
   *string* ← z1 z2;                                                   0468
   RETURN(&string);                                                    0469
   END.
                                                                       0470
(get.jobsoletes) %get the contents of the Obsoletes Documents field
from the JWORK header%
PROCEDURE (string);                                                    0471
   REF string;  LOCAL TEXT POINTER z1, z2;                             0472
   IF FIND jwp1 > /"Obsoletes Document(s): "/  ↑z1 (';/) ↑z2 ←z2 THEN
                                                                       0473
      *string* ← z1 z2                                                 0474
```

```
   ELSE *string* ← NULL;                                                0475
   RETURN(&string);                                                     0476
   END.
                                                                        0477
(getjrfc) %get the contents of the RFC field from the JWORK header%
PROCEDURE (string);                                                     0478
   REF string;  LOCAL TEXT POINTER z1, z2;                              0479
   IF FIND jwp1 > ["RFC# "]   ↑z1 [';] ↑z2 ←z2 THEN                     0480
      *string* ← z1 z2                                                  0481
   ELSE *string* ← NULL;                                                0482
   RETURN(&string);                                                     0483
   END.
                                                                        0484
(getjsubcol) %get the contents of the Sub-Collections field from the
JWORK header%
PROCEDURE (string);                                                     0485
   REF string;  LOCAL TEXT POINTER z1, z2;                              0486
   IF (FIND jwp1 > ["Sub-Collections: "]   ↑z1 [';] ↑z2 ←z2 ) THEN
                                                                        0487
      *string* ← z1 z2                                                  0488
   ELSE *string* ← NULL;                                                0489
   RETURN(&string);                                                     0490
   END.
                                                                        0491
(galjsubcol)PROCEDURE(string, idfnum);                                  01203
   %*galjsubcol-- get all jsubcollection *                              01204
      This is the getj routine which returns the subcollections
      which the document belongs to.                                    01205
      It is a little more complex than most getj routines, because
      the subcollection membership is computed from more than one
      field, and redundant entries must be deleted.                     01206
      The algorithm is:                                                 01207
         Get value of sub-collection field in neader                    01208
         If this is an RFC document, add NWG and NIC to
         subcollections                                                 01209
         If there are any groups on the distribution list for this
         document, enter them into te subcollection list (this uses
         the routine getgpids).                                         01210
      As usual, idfile is the file number of the identfile or zero.
                                                                        01211
      %                                                                 01212
   LOCAL TEXT POINTER z1, z2, z3;                                       01213
   LOCAL STRING tempsr[150], templsr[30];                               01214
   REF string;                                                          01215
   IF (FIND jwp1 > ["Sub-Collections: "]   ↑z1 [';] ↑z2 ←z2 ) THEN
                                                                        01217
      *string* ← z1 z2                                                  01218
   ELSE *string* ← NULL;                                                01238
   IF rfcflg THEN *string* ← "NWG ", *string*;                          01219
   IF rfcflg AND NOT FIND BETWEEN z1 z2(["NIC"]) THEN                   01220
      *string* ← "NIC ", *string*;                                      01221
   IF FIND jwp1 > ["Distribution: "] ['/] ↑z1 THEN                      01222
      BEGIN                                                             01223
         *tempsr* ← NULL;                                               01224
         getgpids($z1, $tempsr, idfnum); %Extract all group
         references from ident list%                                    01225
```

```
        %Now fix up Sub-collection Field to reflect Distribution
        Groups%                                                          01226
           FIND SF(*tempsr*) ↑z1;                                        01227
           WHILE (FIND z1 $NP ↑z2 1$PT ↑z1) DO                           01228
              BEGIN                                                      01229
              *templsr* ← +z2 z1;                                        01230
              IF NOT FIND SF(*string*) [*templsr*] THEN                  01231
                 *string* ← *string*, SP, *templsr*;                     01232
              END;                                                       01233
        END;                                                             01234
     RETURN(&string);                                                    01236
     END.
                                                                         01237
  (getjtitle) %get the contents of the Title field from the JWORK
  header%
  PROCEDURE (string);                                                    0492
     REF string;  LOCAL TEXT POINTER z1, z2;                             0493
     IF FIND jwp1 > ["Title: "] ['"'] -'"' ↑z1 ←z1 ['"'] ↑z2 ←z2 THEN    0494
        *string* ← z1 z2                                                 0495
     ELSE *string* ← NULL;                                               0496
     RETURN(&string);                                                    0497
     END.
                                                                         0498
  (getjupdates) %get the contents of the Updates Documents field from
  the JWORK header%
  PROCEDURE (string);                                                    0499
     REF string;  LOCAL TEXT POINTER z1, z2;                             0500
     IF FIND jwp1 > ["Updates Document(s): "] ↑z1 [';] ↑z2 ←z2 THEN
                                                                         0501
        *string* ← z1 z2                                                 0502
     ELSE *string* ← NULL;                                               0503
     RETURN(&string);                                                    0504
     END.
                                                                         0505
%....catalog manipulation routines....%                                  0506
  (getcacc) %fetch access field from catalog entry%
  PROCEDURE (entrystid, fldstr, ptr1, ptr2);                             0507
     %----------%                                                        0508
     LOCAL retval;                                                       0509
     LOCAL TEXT POINTER z1, z2, z3;                                      0510
     %----------%                                                        0511
     retval ← FALSE;                                                     0512
     makeptr (entrystid, $z3);                                           0513
     IF NOT fndunqtxt ($"AccessList:", $z3, $z1, $z2)                    0514
     OR NOT (FIND z2 > $NP ↑z1 [';] < CH $NP ↑z2) THEN                   0515
        getcend (entrystid, $z1, $z2)                                    0516
     ELSE retval ← TRUE;                                                 0517
     stptset (fldstr, ptr1, ptr2, $z1, $z2);                             0518
     RETURN (retval);                                                    0519
     END.                                                                0520
                                                                         0521
  (getcfn) %fetch filename from catalog entry%
  PROCEDURE (entrystid, fldstr, ptr1, ptr2);                             0522
     %----------%                                                        0523
     LOCAL linkstid, retval;                                             0524
     LOCAL TEXT POINTER tp1, tp2;                                        0525
```

```
   REF fldstr;                                                          0526
   %----------%                                                         0527
   retval ← FALSE;                                                      0528
   IF NOT &fldstr THEN err ();                                          0529
   linkstid ← getsub (entrystid);                                       0530
   IF NOT (FIND SF(linkstid) ["link"] ['(] ↑tp1 ←tp1 [ ')] ↑tp2) THEN   0531
      getcend (entrystid, $tp1, $tp2)                                   0532
   ELSE                                                                 0533
      BEGIN                                                             0534
      lnbfls ($tp1, 0, &fldstr);                                        0535
      *fldstr* ← *fldstr*, ".NLS";                                      0536
      retval ← TRUE;                                                    0537
      END;                                                              0538
   stptset (0, ptr1, ptr2, $tp1, $tp2);                                 0539
   RETURN (retval);                                                     0540
   END.                                                                 0541
                                                                        0542
(getcend) %fetch end of catalog entry%
PROCEDURE (entrystid, ptr1, ptr2);                                      0543
   %----------%                                                         0544
   REF ptr1, ptr2;                                                      0545
   %----------%                                                         0546
   FIND SE(entrystid) ↑ptr1 ↑ptr2;                                      0547
   RETURN;                                                              0548
   END.                                                                 0549
(gtcatent) %locate document in catalogs%
PROCEDURE (docnumber, %opt% catfilename, %or% catfileno,
openformodify);                                                         0550
   %----------%                                                         0551
   LOCAL stid, catalog, fileno, routine, curskpachk;                    0552
   LOCAL STRING catstmtname [10];                                       0553
   LOCAL TEXT POINTER z1;                                               0554
   REF docnumber, catfilename;                                          0555
   %----------%                                                         0556
   IF catfileno THEN %search for statement in open catalog file%        0557
      BEGIN                                                             0558
      %build text pointer to origin statement%                          0559
         stid ← 0;                                                      0560
         stid.stpsid ← origin;                                          0561
         stid.stfile ← catfileno;                                       0562
         makeptr (stid, $z1);                                           0563
      *catstmtname* ← 'J, *docnumber*;                                  0564
      lookup ($z1, $catstmtname, nametyp);                              0565
      RETURN (IF z1 # endfil THEN z1 ELSE FALSE);                       0566
      END;                                                              0567
   IF &catfilename THEN %open and search catalog file%                  0568
      BEGIN                                                             0569
      IF openformodify THEN                                             0570
         BEGIN                                                          0571
         routine ← $openlock;                                           0572
         conjdir (TRUE);                                                0573
         END                                                            0574
      ELSE routine ← $open;                                             0575
      fileno ← 0;                                                       0576
```

```
        curskpachk ← skpachk;                                          0577
        ON SIGNAL ELSE                                                 0578
           BEGIN                                                       0579
           skpachk ← curskpachk;                                       0580
           IF fileno THEN sigclose (fileno := 0);                      0581
           IF openformodify THEN conjdir (openformodify ← FALSE);      0582
           END;                                                        0583
        skpachk ← TRUE;                                                0584
        fileno ← /routine/ (0, jflname (&catfilename));                0585
        IF openformodify THEN fileno ← fileno.stfile;                  0586
        IF NOT (stid ← gtcatent (&docnumber, 0, fileno,
        openformodify)) THEN                                           0587
           BEGIN                                                       0588
           close (fileno := 0);                                        0589
           IF openformodify THEN conjdir (openformodify ← FALSE);      0590
           END;                                                        0591
        skpachk ← curskpachk;                                          0592
        RETURN (stid);                                                 0593
        END;                                                           0594
     %try all catalog files%                                           0595
        IF (stid ← gtcatent (&docnumber, $"Tjcat", 0, openformodify))
                                                                       0596
        OR (stid ← gtcatent (&docnumber, $"Jcat", 0, openformodify))
        THEN                                                           0597
           RETURN (stid);                                              0598
        CASE srval (&docnumber, 10) OF                                 0599
           < 12000: catalog ← $"Jcat0";                                0600
           < 16000: catalog ← $"Jcat1";                                0601
           < 20000: catalog ← $"Jcat2";                                0602
           < 24000: catalog ← $"Jcat3";                               01307
           ENDCASE RETURN (FALSE);                                     0603
        RETURN (gtcatent (&docnumber, catalog, 0, openformodify));     0604
     END.                                                              0605
                                                                       0606
%....field conversion routines....%                                    0607
     (convjsubcol) %convert the contents of the Sub-Collections, Author,
     and Distribution field into a real Sub-Collections field for the
     JWORK header"
     PROCEDURE (idfile);                                               0608
        % convjsubcol documentation                                    0609
           This routine updates the sub-collections field for this
           document.                                                   0610
           The sub-collections membership is computed from the current
           sub-collection, author, and Distribution fields, and redundant
           entries are deleted.  If this is an RFC document, add NWG and
           NIC to subcollections.  If there are any groups on the
           distribution list for this document, enter them into the
           sub-collection list (this uses the routine getgpids).       0611
           %                                                           0612
        LOCAL rfcf;                                                    0613
        LOCAL TEXT POINTER z1, z2, a1, a2;                             0614
        LOCAL STRING                                                   0615
           string[300], tempsr[300], temp1sr[50], temp2sr[50],
           identry[1000];                                              0616
        %---------------%                                             0617
        *string* ← NULL;                                               0618
```

```
   IF (FIND jwp1 > ["Sub-Collections: "] ↑z1 [';] ) THEN                0619
      getgpids ($z1, $string, idfile);                                  0620
   IF rfcf ← ( FIND jwp1 > ["RFC # "] ) THEN                            0621
      IF NOT FIND SF(*string*) ["NWG"] THEN                             0622
         *string* ← *string*, " NWG";                                  0623
   IF rfcf AND NOT FIND BETWEEN z1 z2(["NIC"]) THEN                     0624
      IF NOT FIND SF(*string*) ["NIC"] THEN                             0625
         *string* ← *string*, " NIC";                                   0626
   getjauthor($tempsr);                                                 0627
   %Now fix up Sub-collection Field to reflect authors%                 0628
      FIND SF(*tempsr*) ↑z1;                                            0629
      WHILE (FIND z1 $NP ↑z2 1$PT ↑z1) DO                               0630
         BEGIN                                                          0631
         *temp1sr* ← +z2 z1; %an author%                                0632
         ckident($temp1sr, $identry, idfile); %get his ident entry%
                                                                        0633
         lgetsubcoll($identry, $temp2sr, 0, 0); %get his
         subcollections%                                                0634
         %process each element of his sub-coll list%                   0635
            FIND SF(*temp2sr*) ↑a1;                                     0636
            WHILE (FIND a1 $NP ↑a2 1$PT ↑a1) DO                         0637
               BEGIN                                                    0638
               *temp1sr* ← +a2 a1; %a sub-coll%                         0639
               IF NOT FIND SF(*string*) [*temp1sr*] THEN                0640
                  *string* ← *string*, SP, *temp1sr*;                   0641
               END;                                                     0642
         END;                                                           0643
   IF FIND jwp1 > ["Distribution: "] ['/] ↑z1 THEN  %FIX THIS LATER
   FOR ACTION, INFO DIST LIST%                                          0644
      BEGIN                                                             0645
         *tempsr* ← NULL;                                               0646
         getgpids($z1, $tempsr, idfile); %Extract all group
         references from ident list%                                    0647
         %Now fix up Sub-collection Field to reflect Distribution
         Groups%                                                        0648
            FIND SF(*tempsr*) ↑z1;                                      0649
            WHILE (FIND z1 $NP ↑z2 1$PT ↑z1) DO                         0650
               BEGIN                                                    0651
               *temp1sr* ← +z2 z1;                                      0652
               IF NOT FIND SF(*string*) [*temp1sr*] THEN                0653
                  *string* ← *string*, SP, *temp1sr*;                   0654
               END;                                                     0655
      END;                                                              0656
   IF string.L > 0 THEN setjsubcol($string);                            0657
   RETURN;                                                              0658
   END.
                                                                        0659
(convjdist) %convert Action Distribution and Information-only
Distribution fields into a single Distribution field with comments
appended to recipient idents.%
PROCEDURE;                                                              0660
   LOCAL TEXT POINTER z1,z2, z3, z4;                                    0661
   LOCAL STRING dist[500], idstr[25], comm[300], temp[400];             0662
   IF FIND jwp1 > ["Action Distribution: "] ↑z2 < ['A] CH ↑z1 > z2
   ['/] $NP ↑z2 [';] ↑z4 < CH $NP > ↑z3 THEN %process action list%
                                                                        0663
```

```
        BEGIN %pointers z1 through z4 should be set up as in the
        following example:
           <z1> Action Distribution: / <z2>CHI DIA DCW<z3> ;<z4>  %
                                                                  0664
        *temp* ← z2 z3;  %membership%                             0665
        ST z1 z4 ← NULL;  %delete action list%                    0666
        %add ( [ ACTION ] ) to each ident; must agree with format
        checked in (nls, jnldel, distdl)%                         0667
           FIND SF(*temp*) ↑z1;                                   0668
           WHILE (FIND z1 $NP $(',) $NP ↑z2 1$(LD / '↑ / '& / '-) ↑z1)
           DO                                                     0669
              BEGIN                                               0670
              *idstr* ← +z2 z1;                                   0671
              IF FIND z1 $NP '( ↑z2 ([')] ↑z1 ↑z3 ←z3 /[ENDCHR] ↑z1
              ↑z3) THEN                                           0672
                 *comm* ← z2 z3                                   0673
              ELSE *comm* ← NULL;                                 0674
              *dist* ← *dist*, *idstr*, '(, " [ ACTION ] ", *comm*,
              '), SP;                                             0675
              END;                                                0676
        END;                                                      0677
     IF FIND jwp1 > ["Information-only Distribution: "] ↑z2 < ['I] CH
     ↑z1 > z2 ['/] $NP ↑z2 [';] ↑z4 < CH $NP > ↑z3 THEN %process
     information-only list%                                       0678
        BEGIN %pointers z1 through z4 should be set up as in the
        following example:
           <z1> Information-only Distribution: / <z2>RWW<z3> ;<z4>  %
                                                                  0679
        *temp* ← z2 z3;  %membership%                             0680
        ST z1 z4 ← NULL;  %delete info list%                      0681
        %add ( [ INFO-ONLY ] ) to each ident%                     0682
           FIND SF(*temp*) ↑z1;                                   0683
           WHILE (FIND z1 $NP $(',) $NP ↑z2 1$(LD / '↑ / '& / '-) ↑z1)
           DO                                                     0684
              BEGIN                                               0685
              *idstr* ← +z2 z1;                                   0686
              IF FIND z1 $NP '( ↑z2 ([')] ↑z1 ↑z3 ←z3 /[ENDCHR] ↑z1
              ↑z3) THEN                                           0687
                 *comm* ← z2 z3                                   0688
              ELSE *comm* ← NULL;                                 0689
              *dist* ← *dist*, *idstr*, '(, " [ INFO-ONLY ] ", *comm*,
              '), SP;                                             0690
              END;                                                0691
        END;                                                      0692
     IF dist.L THEN                                               0693
        BEGIN                                                     0694
        FIND jwp1 > [" Author(s): "] [';] ↑z1 ↑z2;                0695
        ST z1 z2 ← " Distribution: /", *dist*, ';;                0696
        END;                                                      0697
     RETURN END.
                                                                  0698
%Procedures concerned with locking/unlocking Journal%             0720
   (jolock)PROC;                                                  0721
      %This procedure checks the global flags which tell the current
      status of te Journal:                                       0722
         Flag 0: (Password jlock): If on Return True, Otherwise False.
```

```
                                                                    0723
        This flag is used to prevent new users fromm entering the
        JOurnal, but does not stop persons who are already in the
        process of using it.                                        0724
     Flag 1: (password jbfil): If on, a file error has been noted
     in one of the JOurnal files.  Discontinnue the current Journal
     Process immediately with a werr.                               0725
                                                                    0726
  %                                                                 0727
  IF jdebug THEN RETURN(FALSE);                                     0728
  IF flagut(1, $tstfg) THEN                                         0729
     %Some one has found a bad file somewhere%                      0730
     werr($"Journal File System Error--Call NIC Center");           0731
  RETURN(flagut(0, $tstfg));
  END.
                                                                    0732
(lockjo)PROC(type);                                                 0733
  %Type = 1 for bad file lock (flag 1), and 0 for jlock (flag 0);
                                                                    0734
  %                                                                 0735
  IF type > 1 THEN err($"Illegal Flag Number");                     0736
  flagut(type, $setfg);                                             0737
  specttyout(0, $"Journal Locked");                                 0738
  specttyout(308, $"Journal Locked");                               0739
  RETURN;                                                           0740
  END.                                                              0741
(unlkjo)PROC(type);                                                 0742
  %Type indates flag to be unlocked... -1 means all%                0743
  IF type NOT IN [-1, 1] THEN err($"Illegal Flag Number");          0744
  IF type # 1 THEN flagut(0, $rstfg);                               0745
  IF type # 0 THEN flagut(1,$rstfg);                                0746
  RETURN;                                                           0747
  END.                                                              0748
%....Journal submission....%                                        0749
  (jsubmit) %do the Journal submission or secondary distribution%
  PROCEDURE;                                                        0750
     %This procedure submits an itemm to the Journal by creating a
     properly formatted file in the directory <tejournal>.%         0751
     LOCAL  rnumstid, numstid, fl, traping, capsav, connflag, stid,
     orgstr;                                                        0752
     LOCAL TEXT POINTER z1, z2, z3, z4, z5, z6, where;              0767
     LOCAL STRING                                                   0754
        number[20],      %journal number%                          0755
        rfcnum[20],      %RFC number%                              0756
        itemtype[20],    %item type%                               0757
        linkaddr[150],   %link to location for link insert%        0758
        fnamestr[150],   %file name string%                        0759
        goodids[500],    %list of good idents%                     0760
        badids[500],     %list of bad idents%                      0761
        workstr[500],    %work string%                             0762
        accesslist [1000], %access list%                          01197
        dpassw[15],      %password string%                         0763
        dirnam[50],      %directory name%                          0764
        dfilstr[150],    %tejournal file name%                     0765
        mfname[150];    %message file name%                        0766
     REF fl, orgstr;                                               0753
     jolock(); %Check to make sure it will work -- does not return
```

```
unless things are cool%                                                 0768
%set stid's to null%                                                    0769
   rnumstid ← numstid ← orgstid;                                        0770
capsav ← 0;                                                             0771
traping ← connflag ← FALSE;                                             0772
&orgstr ← 0;                                                           01289
ON SIGNAL                                                               0773
   ELSE                                                                 0774
      BEGIN %close any open files%                                      0775
      ON SIGNAL ELSE;                                                   0776
      condir(FALSE);                                                    0777
      IF numstid.stfile THEN close(numstid.stfile := 0);                0778
      IF rnumstid.stfile THEN close(rnumstid.stfile := 0);              0779
      IF traping := FALSE THEN notrapcc(capsav);                        0781
      IF connflag := FALSE THEN %connect back to user's
      directory%                                                        0782
         condir($dirnam,$dpassw,@dirnam,$dpassw);                       0783
      % Check orgstr; if empty, simply release string.  If not,
      reset the origin statement to its original state. %              01290
         IF &orgstr THEN                                               01291
            BEGIN                                                      01292
            IF orgstr.L THEN                                           01293
               BEGIN                                                   01294
               % Restore original origin statement. %                  01295
                  ST jwp1 ← *orgstr*;                                  01296
               END;                                                    01297
            freestring(&orgstr, $dspblk);                              01298
            END;                                                       01299
      END;                                                              0784
IF NOT idfno THEN idfno ← loaidfil();                                   0785
% Get string to be used for origin statement restoration. %            01300
   &orgstr ← getstring(2000, $dspblk);                                 01301
   *orgstr* ← SF(jwp1) SE(jwp1);                                       01302
%check for non-null source%                                            01249
   stid ← getsub(jwp1);                                                01250
   IF getnxt(stid) = endfil AND ( FIND SF(stid) *nullsource*
   ENDCHR ) THEN                                                       01251
      werr($"Nothing to send!");                                       01252
%check ident lists%                                                     0786
   getjauthor($workstr);                                                0787
      IF *workstr* # *initsr* THEN %must check it%                      0788
         BEGIN                                                          0789
         *goodids* ← *workstr*;                                         0790
         ckidlist($goodids, $badids, idfno);                            0791
         IF badids.L THEN %tell user the bad news%                      0792
            BEGIN                                                       0793
            IF goodids.L THEN setjauthor($goodids)                      0794
            ELSE setjauthor($initsr);                                   0795
            *badids* ← "Illegal ident(s) (mail not sent): ",
            *badids*;                                                   0796
            werr($badids);                                              0797
            END;                                                        0798
         IF *goodids* # *workstr* THEN setjauthor($goodids);            0799
         END;                                                           0800
   getjaction($workstr);                                                0801
      IF workstr.L THEN %must check it%                                 0802
```

```
            BEGIN                                                   0803
            *goodids* ← *workstr*;                                  0804
            ckidlist($goodids, $badids, idfno);                     0805
            IF badids.L THEN %tell user the bad news%               0806
               BEGIN                                                0807
               setjaction($goodids);                                0808
               *badids* ← "Illegal ident(s) [mail not sent]: ",
               *badids*;                                            0809
               werr($badids);                                       0810
               END;                                                 0811
            IF *goodids* # *workstr* THEN setjaction($goodids);     0812
            END;                                                    0813
         getjinfo($workstr);                                        0814
            IF workstr.L THEN %must check it%                       0815
            BEGIN                                                   0816
            *goodids* ← *workstr*;                                  0817
            ckidlist($goodids, $badids, idfno);                     0818
            IF badids.L THEN %tell user the bad news%               0819
               BEGIN                                                0820
               setjinfo($goodids);                                  0821
               *badids* ← "Illegal ident(s) [mail not sent]: ",
               *badids*;                                            0822
               werr($badids);                                       0823
               END;                                                 0824
            IF *goodids* # *workstr* THEN setjinfo($goodids);       0825
            END;                                                    0826
      %set up strings for later use%                                0827
         getjrfc($rfcnum);                                          0828
         getjnumber($number);                                       0829
            IF NOT rfcnum.L AND NOT number.L THEN  %kludge for now
            until fix background guy%                               0830
            BEGIN                                                   0831
            *number* ← "1234";                                      0832
            setjnumber($"1234");                                    0833
            END                                                     0834
            ELSE IF *number* # "1234" THEN %check it%               0835
            BEGIN.                                                  0836
            getjauthor($workstr);                                   0837
            IF NOT FIND SF(*workstr*) [*initsr*] THEN               0838
               *workstr* ← *initsr*, SP, *workstr*;                 0839
            conjdir(TRUE);                                          0840
            numstid ← openlock(0, jflname($"tcnumbers"));           0841
            IF rfcnum.L THEN %check rfc number and journal number%
                                                                    0842
               BEGIN                                                0843
               rnumstid ← openlock(0, jflname($"Rfcnumbers"));      0844
               ckrfcnum(1, $rfcnum, $number, $workstr, rnumstid,
               numstid);                                            0845
               END                                                  0846
            ELSE ckcnum(1, $number, $workstr, numstid);             0847
            conjdir(FALSE);                                         0848
            END;                                                    0849
         getjlink($linkaddr);                                       0850
         getjtype($itemtype);                                       0851
      %fill in time and date (of submission) fields%                0852
         *datesr* ← NULL;                                           0853
```

```
    !gtad;                                                          0854
    dtfrmt(r1, $datesr);                                            0855
    IF NOT FIND jwp1 > [";;;;"] [ "(J" / "(j" ] [';] ↑p1 ←p1
    ["Author(s):"] ['/] ↑p3 [';] ↑p4 ←p4 THEN                       0856
       werr($"Journal Header improperly formatted [mail not
       sent]");                                                     0857
    IF rfcnum.L THEN                                                0858
       ST jwp1 ← SF(p1) p1, SP, *datesr*, "; .HJOURNAL=", '",
       "NWG/RFC# ", *rfcnum*, ".SPLIT;",  p3 p4, SP, *datesr*, "
       ", *number*, '",  p1 SE(p1)                                  0859
    ELSE                                                            0860
       ST jwp1 ← SF(p1) p1, SP, *datesr*, "; .HJOURNAL=", '", p3
       p4, SP, *datesr*, "  ", *number*, '",  p1 SE(p1);           0861
%build access list if private document%                             0862
    IF rdprvsts (jwp1.stfile) = $psprivate THEN                     0705
       BEGIN                                                        01195
       accesslist.L ← 0;                                            0707
       clnidist (getjaction ($workstr), $accesslist);               0708
       clnidist (getjinfo ($workstr), $accesslist);                 0709
       clnidist (getjauthor ($workstr), $accesslist);               0710
       clnidist (getjclerk ($workstr), $accesslist);                0711
       setjaxcess ($accesslist);                                    0713
       END;                                                         01196
%convert subcollections and distribution list -- should be done
when tejournal file is loaded by background%                        0864
    convjdist();                                                    0865
    convjsubcol(idfno);                                             0866
%decide between storage as message or file%                         0867
    IF FIND jwp1 > [";;;; "] ↑p1 CH ↑p2 THEN                        0868
       IF rdprvsts (jwp1.stfile) = $psprivate                       01199
       OR getnxt (getnxt (jworkstid)) # endfil                      01200
       % OR NOT jtypchk (2000, jworkstid) % THEN                    01198
          ST p1 p2 ← 'F                                             0869
       ELSE ST p1 p2 ← 'M;                                          01202
%conn. to TEJOURNAL directory and update the workfile into it%
                                                                    0870
    IF NOT number.L OR *number* = "1234" THEN  %build
    deferred-number file name%                                      0871
       BEGIN                                                        0872
       FIND SF(*datesr*) [1$D] ↑z2 < D ↑z1 > [':] ↑z5 <CH ↑z4 $D
       ↑z3 z5 > $D ↑z6;                                             0873
       *dfilstr* ← 'D, z1 z2, z3 z4, z5 z6, *initsr*, ".NLP";       0874
       END                                                          0875
    ELSE  %build pre-assigned number file name%                     0876
       *dfilstr* ← 'J, *number*, ".NLN";                            0877
    *dirnam* ← NULL;                                                0878
    &f1 ← flntadr(jworkstid.stfile);                                0879
    jfnstr(f1.florig, $fnamestr);                                   0880
    traping ← TRUE;                                                 0881
    capsav ← trapcc();                                              0882
    connflag ← TRUE;                                                0883
    condir(IF jdebug THEN $"IRBY" ELSE $"TEJOURNAL",IF jdebug THEN
    $"BLI" ELSE $jnlpsw,$dirnam,$dpassw);                           0884
    updtfl(jworkstid.stfile, newversion, $dfilstr);                 0885
    condir($dirnam,$dpassw,$dirnam,$dpassw);                        0886
    connflag ← FALSE;                                               0887
```

```
    IF traping:=FALSE THEN notrapcc(capsav);                          0888
    fl.flnoclos ← FALSE;                                              0890
    close(jworkstid.stfile);                                          0891
    jworkstid ← 0;                                                    0892
    % Dispose of orgstr space. %                                     01303
       freestring(&orgstr:=0, $dsphlk);                              01304
    % Zap the Journal workfile. %                                    01305
       ON SIGNAL ELSE                                                 0893
          BEGIN %must create it%                                      0894
          ON SIGNAL ELSE;                                             0895
          jworkstid ← openwk(0, $fnamestr);                           0896
          GOTO jsubok;                                                0897
          END;                                                        0898
       jworkstid ← orgstid;                                           0899
       jworkstid.stfile ← open(0, $fnamestr);                         0900
       resetf(jworkstid.stfile);                                      0901
       (jsubok):  %initialize the file%                               0902
       &fl ← flntadr(jworkstid.stfile);                               0903
       fl.flnoclos ← TRUE;                                            0904
       initjwork();  %zap the Journal workfile%                       0905
       FIND SF(jworkstid) ↑jwpl;                                      0906
ON SIGNAL                                                             0909
    ELSE                                                              0910
       BEGIN %close any open files%                                   0911
       ON SIGNAL ELSE;                                                0912
       conjdir(FALSE);                                               01306
       IF numstid.stfile THEN close(numstid.stfile := 0);             0913
       IF rnumstid.stfile THEN close(rnumstid.stfile := 0);           0914
       IF idfno THEN close(idfno := 0);                               0915
       IF traping := FALSE THEN notrapcc(capsav);                     0916
       END;                                                           0917
IF number.L AND *number* # "1234" THEN                                0907
    BEGIN                                                             0908
    conjdir(TRUE);                                                    0918
    IF rfcnum.L THEN                                                  0919
       BEGIN %release RFC number%                                     0920
       usedcnum($rfcnum, rnumstid);                                   0921
       END;                                                           0922
    usedcnum($number, numstid); %release used journal number%         0923
    conjdir(FALSE);                                                   0924
    END;                                                              0925
IF linkaddr.L THEN                                                    0926
    BEGIN                                                             0927
    CASE *itemtype*[1] OF                                             0928
       = 'H: *mfname* ← "(Hard Copy -- Journal, ", *number*, ",)";
                                                                      0929
       ENDCASE *mfname* ← "( Journal, ", *number*, ", 1:w)" ;         0930
    %now insert the link%                                             0931
       FIND SF(*linkaddr*) ↑z1 SE(*linkaddr*) ↑z2;                    0932
       caddexp($z1, $z2, lda(), $where);                              0933
          %where should the starting point of the expression
          be???? we leave it undefined%                              01308
       IF where = endfil THEN                                         0934
          BEGIN                                                       0935
          *linkaddr* ← "Unable to insert link ", *mfname*, " at ",
          *linkaddr*;                                                 0936
```

```
            dismes(1, $linkaddr);                                          0937
            END                                                            0938
         ELSE                                                              0939
            BEGIN                                                          0940
            FIND SF(*mfname*) ↑z1 SE(*mfname*) ↑z2;                        0941
            dismes(1, $linkaddr);                                          0942
            cinstex($where, $z1, $z2, TRUE);                               0943
            END;                                                           0944
      END;                                                                 0945
   dismes(1, $"Completed");                                                0946
   IF idfno THEN close(idfno := 0);                                        0947
   IF numstid.stfile THEN close(numstid.stfile := 0);                      0948
   IF rnumstid.stfile THEN close(rnumstid.stfile := 0);                    0949
   RETURN;                                                                 0950
   END.                                                                    0951
(secdist)PROCEDURE(number, distlist, actionflag);  %execute forward
journal item request%                                                      01309
   %Accepts a string containing a catalog number, and a string
   containing an identlist and a flag (true if action, false if
   info-only).                                                             01310
      Distributes the indicated Journal document to the persons on
      the list%                                                            01311
   LOCAL char, jcstid, f1, traping, capsav, connflag;                      01312
   REF number, distlist, f1;                                               01315
   LOCAL TEXT POINTER z1, z2, z3, z4;                                      01313
   LOCAL STRING                                                            01452
      badids[500],     %list of bad idents%                               01459
      dpassw[15],      %password string%                                  01462
      dirnam[50],      %directory name%                                   01463
      dfilstr[150],    %tejournal file name%                              01464
      idstr[25],       %to hold an ident%                                 01469
      comm[300],       %to hold the comment associated with an ident%
                                                                          01468
      fnamestr[150],   %file name string%                                 01496
      tempsr[400];     %temporary work string%                            01467
   jcstid ← orgstid;                                                       01316
   capsav ← 0;                                                             01474
   traping ← connflag ← FALSE;                                             01473
   ON SIGNAL                                                               01475
      ELSE                                                                 01476
         BEGIN %close any open files%                                      01477
         ON SIGNAL ELSE;                                                  01478
         condir(FALSE);                                                   01479
         IF traping := FALSE THEN notrapcc(capsav);                       01482
         IF connflag := FALSE THEN %connect back to user's
         directory%                                                       01483
            condir(@dirnam,$dpassw,$dirnam,$dpassw);                      01484
         close(jcstid.stfile := 0);                                       01317
         END;                                                             01495
   IF NOT (jcstid ← gtcatent (&number, 0, 0, FALSE)) THEN                  01318
      werr ($"No such document");                                         01319
   IF NOT vrprvacc (0, 0, jcstid, $initsr) THEN                            01320
      werr ($"Private document; access denied to you");                   01321
   IF NOT idfno THEN idfno ← loaidfil();                                   01342
   %check distribution list%                                               01357
```

```
    ckidlist($distlist, $badids, idfno);                                    01361
    IF badids.L THEN %tell user the bad news%                               01362
       BEGIN                                                                01363
       *badids* ← "Illegal ident(s) /mail not forwarded/: ",
       *badids*;                                                            01365
       werr($badids);                                                       01366
       END;                                                                 01367
 %put in ACTION or INFO-only designation%                                   01409
    IF actionflag THEN %process as action list%                             01373
       *dirnam* ← " / ACTION / "                                            01471
    ELSE %process as information-only list%                                 01388
       *dirnam* ← " / INFO-ONLY / ";                                        01472
    *tempsr* ← *distlist*;                                                  01375
    *distlist* ← NULL;                                                      01470
    %add ( / ACTION / )  or ( /INFO-ONLY/ ) to each ident; must
    agree with format checked in (nls, jnldel, distdl)%                     01377
       FIND SF(*tempsr*) ↑z1;                                               01378
       WHILE (FIND z1 $NP $(',) $NP ↑z2 1$(LD / '↑ / '& / '-) ↑z1)
       DO                                                                   01379
          BEGIN                                                             01380
          *idstr* ← +z2 z1;                                                 01381
          IF FIND z1 $NP '( ↑z2 (/')/ ↑z1 ↑z3 ←z3 //ENDCHR/ ↑z1
          ↑z3) THEN                                                         01382
             *comm* ← z2 z3                                                 01383
          ELSE *comm* ← NULL;                                               01384
          *distlist* ← *distlist*, *idstr*, '(, *dirnam*, *comm*,
          '), SP;                                                           01385
          END;                                                              01386
    %terminate with semicolon%                                              01497
       *distlist* ← *distlist*, '; ;                                        01498
 ST jworkstid ← ";;;; S ", SF(jcstid) SE(jcstid);                           01326
 z2 ← getsub(jcstid);                                                       01327
 z2 ← ccopsta(jworkstid, levdown, z2, FALSE, 0);                            01328
 z2 ← cis(z2, &distlist, $sucdir);                                          01330
 cis(z2, $initsr, $sucdir);                                                 01331
 %update to TEJOURNAL%                                                      01332
    *dfilstr* ← 'S, *number*, ".NLN";                                       01333
    *dirnam* ← NULL;                                                        01334
    &fl ← flntadr(jworkstid.stfile);                                        01421
    jfnstr(fl.florig, $fnamestr);                                           01422
    traping ← TRUE;                                                         01423
    capsav ← trapcc();                                                      01424
    connflag ← TRUE;                                                        01425
    condir(IF jdebug THEN $"IRBY" ELSE $"TEJOURNAL",IF jdebug THEN
    $"BLI" ELSE $jnlpsw,$dirnam,$dpassw);                                   01426
    updtfl(jworkstid.stfile, newversion, $dfilstr);                         01427
    condir($dirnam,$dpassw,$dirnam,$dpassw);                                01428
    connflag ← FALSE;                                                       01429
    IF traping:=FALSE THEN notrapcc(capsav);                                01430
    fl.flnoclos ← FALSE;                                                    01431
    close(jworkstid.stfile);                                                01432
    jworkstid ← 0;                                                          01433
    % Zap the Journal workfile. %                                          01436
       ON SIGNAL ELSE                                                       01437
          BEGIN %must create it%                                            01438
          ON SIGNAL ELSE;                                                   01439
```

```
            jworkstid ← openwk(0, $fnamestr);                          01440
            GOTO jsubok;                                               01441
            END;                                                       01442
         jworkstid ← orgstid;                                          01443
         jworkstid.stfile ← open(0, $fnamestr);                        01444
         resetf(jworkstid.stfile);                                     01445
         (secdok):  %initialize the file%                              01446
         &fl ← fintadr(jworkstid.stfile);                              01447
         fl.flnoclos ← TRUE;                                           01448
         initjwork();  %zap the Journal workfile%                      01449
         FIND SF(jworkstid) ↑jwpl;                                     01450
      dismes(1, $"Completed");                                         01339
      close(jcstid.stfile := 0);                                       01340
      RETURN END.                                                      01341

%....Network Journal Procedures....%                                    0952
   (njs) %network journal submission%
   PROCEDURE (acs);                                                    0953
      %----------%                                                     0954
      LOCAL ct, nxtstid, substid, injfn, titlestid, boundstid, tally,
      tailstrt, tailend;                                               0955
      LOCAL STRING                                                     0956
         infile [40], control [200], authlist [200], distlist (200),
         clerk [20], title [200], errstr [500], level [30], workfile
         [40];                                                         0957
      LOCAL TEXT POINTER tp1, tp2, tp3, tp4, tp5;                      0958
      REF acs;                                                         0959
      %----------%                                                     0960
      %avoid scheduler pit%                                            0961
         r1 ← 400000B;                                                 0962
         r2 ← 303B;                                                    0963
         ! JSYS spriw;                                                 0964
      %open, read, and close control file%                             0965
         ON SIGNAL ELSE                                                0966
            hiortn (cat, $"Control file error: ", sysmsg, 0);          0967
         r1 ← acs [7]; ! HRRZS 1; injfn ← r1 :                         0968
         if not sysopen (injfn, read, chrtyp, $errstr) THEN            0969
            hiortn (cat, $"Control file can't be opened: ", $errstr,
            0);                                                        0970
         IF NOT isread (injfn, $control, CR) THEN                      0971
            hiortn (cat, $"Control file can't be read", 0, 0);         0972
         sysclose (injfn, $errstr);                                    0973
      %determine conversion type%                                      0974
         ct ← 'S;                                                      0975
         IF FIND SF(*control*) [';] ↑tp1 ←tp1 ↑tp2 CH ↑tp3 tp2 THEN    0976
            BEGIN                                                      0977
            CASE (ct ← READC .A (177B-SP)) OF                          0978
               ='S, %insert sequential%                                0979
               ='H, %heuristic w/o prev right justfiy%                 0980
               ='J, %heuristic w/ prev right justify%                  0981
               ='A, %insert assembler w/ structure%                    0982
               ='M: %insert assembler w/o structure%                   0983
                  NULL;                                                0984
               ENDCASE                                                 0985
                  hiortn (cat, $"Valid conversion types are A, M, S, H,
                  and J", 0, 0);                                       0986
```

```
      ST tp1 tp3 ← NULL;                                              0987
      END;                                                            0988
   ct ← 'S; %force sequential conversion for now%                     0989
%verify and save clerk, author(s), addressee(s)%                      0990
   ON SIGNAL ELSE                                                     0991
      hiortn (cat, $"Ident verification error: ", sysmsg, 0); 0992
   IF NOT FIND SF(*control*) $(SP/',) ↑tp1 1$(LD/'-) ↑tp2 ['/]
   ↑tp3 ←tp3 ↑tp4 [CR] ↑tp5 ←tp5 THEN                                 0993
      hiortn (cat, $"Ident list format error", 0, 0);                 0994
   *clerk* ← + tp1 tp2;                                               0995
   *initsr* ← *clerk*;                                                0996
   *authlist* ← + tp1 tp3;                                            0997
   *distlist* ← + tp4 tp5;                                            0998
   IF NOT njsverids ($clerk, $authlist, $distlist, $errstr)          0999
      THEN hiortn (cat, $"No such ident(s): ", $errstr, 0);          01000
   hioco (ok, $acs);                                                 01001
%fetch name of sequential text file%                                 01002
   ON SIGNAL ELSE                                                    01003
      hiortn (cat, $"Work file init. error: ", sysmsg, 0);           01004
   r1 ← acs [7]; ! HRRZS 1; injfn ← r1;                              01005
   jfnstr (injfn, $infile);                                          01006
%initialize work file%                                               01007
   *workfile* ← "<SYSTEM>[SEND-MAIL].", *initsr*, ";1;P707000";
                                                                     01008
   ON SIGNAL ELSE                                                    01009
      BEGIN %must create it%                                         01010
      ON SIGNAL ELSE                                                 01011
         hiortn (cat, $"Work file init. error: ", sysmsg, 0);
                                                                     01012
      jworkstid ← openwk (0, $workfile);                             01013
      GOTO okay;                                                     01014
      END;                                                           01015
   jworkstid ← orgstid;                                              01016
   jworkstid.stfile ← open (0, $workfile);                           01017
   initjwork ();                                                     01018
   FIND SF(jworkstid) ↑jwp1;                                         01019
   (okay): ON SIGNAL ELSE                                            01020
      hiortn (cat, $"Work file init. error: ", sysmsg, 0);           01021
   setjauthor ($authlist);                                           01022
   setjaction ($distlist);                                           01023
%insert text of message/file%                                        01024
   IF (substid ← getsub (jworkstid)) # jworkstid THEN                01025
      cdelgro (substid  getail (substid), FALSE, 0);                 01026
   level.L ← 0;                                                      01027
   CASE ct OF                                                        01028
      ='S: %insert sequential%                                       01029
         inseq (jWorkstid, $level, $infile, tenfil);                 01030
      ='H: %heuristic insert sequential w/o prev right justify%
                                                                     01031
         inseqn (jWorkstid, $level, $infile, FALSE);                 01032
      ='J: %heuristic insert sequential w/ prev right justify%
                                                                     01033
         inseqn (jWorkstid, $level, $infile, TRUE);                  01034
      ='A: %insert assembler w/ Structure%                           01035
         inseq (jWorkstid, $level, $infile, assfil);                 01036
      ='M: %insert assembler W/o Structure%                          01037
```

```
            inseq (jworkstid, $level, $infile, macfil);
         ENDCASE;
   %extract the title (if any)%
      titlestid ← boundstid ← tailstrt ← tally ← 0;
      IF ((nxtstid ← getnxt (jworkstid)) # endfil) THEN
         WHILE ((nxtstid # jworkstid) AND ((tally ← tally+1) <= 15))
         DO
            BEGIN
            IF FIND SF(nxtstid) ↑tp1 [':] ↑tp2 ←tp2 $SP ↑tp3
            ([CR/ECL] < CH / SE(nxtstid)) $SP ↑tp4 THEN
               IF (tp2 [1] - tp1 [1]) <= title.M THEN
                  BEGIN
                  *title* ← + tp1 tp2;
                  IF *title* = "RE"
                  OR *title* = "TITLE"
                  OR *title* = "SUBJECT" THEN
                     BEGIN
                     titlestid ← nxtstid;
                     *title* ← tp3 tp4;
                     IF title.L THEN setjtitle ($title);
                     END;
                  END;
            IF FIND SE(nxtstid) "- - - -" THEN
               BEGIN
               boundstid ← nxtstid;
               WHILE (NOT getftl (nxtstid)) DO
                  BEGIN
                  nxtstid ← getsuc (nxtstid);
                  IF FIND SE(nxtstid) "-------" THEN
                     BEGIN
                     tailstrt ← nxtstid;
                     WHILE (NOT getftl (nxtstid)) DO
                        nxtstid ← getsuc (nxtstid);
                     tailend ← nxtstid;
                     EXIT LOOP 2;
                     END;
                  END;
               EXIT LOOP;
               END;
            IF NOT FIND SF(nxtstid) [PT] THEN
               BEGIN
               boundstid ← nxtstid;
               EXIT LOOP;
               END;
            nxtstid ← getsuc (nxtstid);
            END;
      ON SIGNAL ELSE GOTO done;
      IF tailstrt THEN
         cdelgro (tailstrt, tailend, FALSE, 0);
      IF boundstid THEN
         cdelgro (getnxt (jworkstid), boundstid, FALSE, 0);
      IF titlestid THEN cdelsta (titlestid, FALSE, 0);
      (done): ON SIGNAL ELSE
         nioprn (cat, $"JWORK file init. error: ", sysmsg, 0);
   %null document?%
      IF getnxt (jworkstid) = endfil THEN
```

```
         cis (jworkstid, $nullscurce, $"d");
      %journalize%
         ON SIGNAL ELSE
            hiortn (cat, $"Submission error: ", sysmsg, 0);
         jsubmit ();
         hiortn (ok, $"Journalized", 0, 0);
         RETURN;
         END.
(njsverids) %verify ident lists%
PROCEDURE (clerk, authlist, distlist, errlst);
      %----------%
      LOCAL idfileno;
      LOCAL STRING ident [30];
      LOCAL TEXT POINTER tp1, tp2;
      REF clerk, authlist, distlist, errlst;
      %----------%
      idfileno ← open (0, jflname ($"Identfile"));
      ON SIGNAL ELSE IF idfileno THEN
         sigclose (idfileno := 0);
      errlst.L ← 0;
      %verify clerk%
         IF NOT oldid (&clerk, idfileno) THEN
            *errlst* ← *errlst*, *clerk*, SP;
      %verify authors%
         FIND SF(*authlist*) ↑tp2;
         WHILE (FIND tp2 > $(SP/',) ↑tp1 1$(LD/'-) ↑tp2) DO
            BEGIN
            *ident* ← tp1 tp2;
            IF NOT oldid ($ident, idfileno) THEN
               *errlst* ← *errlst*, *ident*, SP;
            END;
      %verify addressees%
         FIND SF(*distlist*) ↑tp2;
         WHILE (FIND tp2 > $(SP/',) ↑tp1 1$(LD/'-) ↑tp2) DO
            BEGIN
            *ident* ← tp1 tp2;
            IF NOT oldid ($ident, idfileno) THEN
               *errlst* ← *errlst*, *ident*, SP;
            END;
      IF errlst.L THEN BUMP DOWN errlst.L;
      close (idfileno := 0);
      RETURN (NOT errlst.L);
      END.

(hiortn) %return to calling fork%
PROCEDURE (outcome, hdr, body, trlr);
      %----------%
      LOCAL STRING msg [500];
      REF hdr, body, trlr;
      %----------%
      msg.L ← 0;
      IF &hdr THEN *msg* ← *msg*, *hdr*;
      IF &body THEN *msg* ← *msg*, *body*;
      IF &trlr THEN *msg* ← *msg*, *trlr*;
      dismes (2, $msg);
      RETURN;
```

```
      *msg* ← *msg*, 0;                                                    01149
      r2 ← chbptr (0) + $msg;                                              01150
      r1 ← outcome;                                                        01151
      ! haltf;                                                             01152
      END.                                                                 01153
   (hioco) %co-routine return to calling fork%
   PROCEDURE (outcome, acs);                                               01154
      %----------%                                                         01155
      LOCAL STRING filename [40], error [200];                             01156
      REF acs;                                                             01157
      %----------%                                                         01158
      *filename* ← "NJS.SRC";                                              01159
      acs [7] ← sgtjfn (100000000000B, $filename, $error);                 01160
      RETURN;                                                              01161
      r1 ← outcome;                                                        01162
      r2 ← 0;                                                              01163
      r4 ← $acs [7]; ! MOVE 3,7;                                           01164
      ! haltf;                                                             01165
      ! EXCH 3,7; ! MOVEM 3,(4);                                           01166
      RETURN;                                                              01167
      END.                                                                 01168
% miscellaneous  %                                                        01169
   (jtypchk) PROC (maxchars, jstid); % check number of chars in a
   jworkfile  %                                                            01170
      % jstid is statement preceeding the body of the submssion %          01171
      % Returns TRUE if <= max %                                          01172
      LOCAL  stid, size;                                                   01173
      size ← 0;                                                            01174
      stid ← jstid;                                                        01175
      WHILE (stid ← getnxt(stid)) # endfil AND size <= maxchars DO         01176
         size ← size + getstsize(stid);                                    01177
      RETURN(IF size > maxchars THEN FALSE ELSE TRUE);                     01178
      END.
                                                                           01179
   (jlog)PROC(number, typest, identsr);                                    01180
      %Type message to Journal Logging tty.                                01181
         Number = Number                                                   01182
         typest is a string which describes use of number                  01183
         identsr is the ident of author, etc.                              01184
      %                                                                    01185
      REF number, typest, identsr;                                         01186
      LOCAL STRING tempsr[200];                                            01187
      *tempsr* ← *number*, SP, *typest*, SP, *identsr*, SP;                01188
      !JSYS gtad;                                                          01189
      dtfrmt(r1, $tempsr); %Add the date/time%                             01190
      specttyout(10B, $tempsr);                                            01191
      RETURN;                                                              01192
      END.                                                                 01193
FINISH                                                                     01194
```

(MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)
(MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)CSENDMAIL (MLK)CSENDMAIL

```
< NLS, CSENDMAIL.NLS;27, >, 4-OCT-74 13:21 CHI ;;;;( NLS,
CSENDMAIL.NLS;23, ), 10-JUN-74 13:23 CHI ;
FILE csendmail % 110 to <rel-nls>csendmail % % (110.sav,)  (rel-nls,
csendmail.rel,) %                                                    02
%....Declarations....%                                               03
   REGISTER r1=1, r2=2, r3=3, r4=4, r5=5, r6=6, r7=7;                04
   DECLARE    %for Journal invocation from another (e.g. FTP) process%
                                                                     05
      ok=1,          % successful return %                          06
      cat=0,         % catastrophic error return %                  07
      submission=1, % journal submission %                         08
      leaving=-1,    % file leaving system %                        09
      enterin#=0;    % file entering system %                      010
   DECLARE STRING nullsource = "[No content was specified!]";     01253
%....Initialize JWORK file....%                                     011
   (initjwork) %Initialize  JWORK file%
   PROCEDURE;                                                       012
      % DOCUMENTATION for initjwork:                               013
         This procedure initializes the JWORK file, which is the work
         file for sending or forwarding Jounal items.              014
            The JWORK file is named a concatenation of
            "[send-mail].IDENT , where IDENT is the ident of the person
            using NLS.                                             015
            The JWORK file is created in the logged in directory by
            xjloaworfil.  It is assumed that the file is NULL (has only
            simple origin statement) at the time this routine is
            called.                                                016
         This routine initializes the origin statement as the Journal
         header.                                                   017
            Replace existing origin statement with:                018
               Journal Number                                      019
                  (a dummy meaning deferred number assingment)     020
               Author                                              021
                  (ident of user)                                  022
               Message flag                                        023
                  (this will be changed when user specifies source of
                  item being sent or forwarded)                    024
               Clerk                                               025
                  (ident of user)                                  026
               Journal Directives                                  027
               2 EOL's (for delimiting comments later)             028
         Finally, it inserts a dummy message as statement 1.       029
         %                                                         030
      LOCAL stid;                                                  031
      %--------------------%                                       032
      %check jworkstid and setup jwpl%                             033
         jworkstid.stpsid ← origin;  %just for safety's sake%      034
         IF NOT goodrng(jworkstid) THEN                            035
            err(#"Journal Work File Cannot be Initialized");       036
         FIND SF(jworkstid) ↑jwpl;                                 037
      %make sure name delimiters are right%                      01499
         csetnsta(jworkstid, '(, ') );                           01500
      %set up Journal Header%                                      038
         ST jwpl← ";;;; F (J) ; Author(s): /", *initsr*, ';,       039
            " Clerk: ", *initsr*, ';,                              040
            %Journal Formatting Directives followed by two EOL's%  041
```

```
            " .IGD=0; .SNF=HJRM; .RM=HJRM-7;  .PN=-1; .YBS=1;
            .PES;", EOL, EOL;                                          042
      %insert dummy message%                                           043
         IF (stid ← getsub(jworkstid)) NOT= jworkstid THEN             044
            BEGIN %there is substructure, delete it%                   045
            cdelgro(stid, getail(stid), FALSE, 0);                     046
            END;                                                       047
         cis(jworkstid, $nullsource, $"d");                            048
      RETURN END.
                                                                       049
%....Set contents of Journal header fields....%                        050
   % DOCUMENTATION FOR setj routines:                                  051
      These routines are used to set values of fields in the Journal
      header statement in JWORK file.                                  052
      They assume JWP1 is set up to point to the first character of the
      origin statement of the JWORK file and are, in general, called
      with a single parameter, the address of a STRING which contains
      the value to which the field is to be set.                       053
      They subsequently search the header statement for the field (or
      the proper location of the field if the field is optional and
      non-present), and replace the existing field with the string (or
      create the field if necessary).                                  054
      Where there are a number of optional fields immediately adjacent
      to each other, the search algorithm uses a list of alternatives,
      searching for the first preceding field first, and if that fails,
      the second preceding field, and if that fails.....               055
      Setj routines which do unusual things will be separately
      documented.                                                      056
      %                                                                057
   (setjauthor) %set the author field of JWORK file%
   PROCEDURE (authorstr);                                              058
      LOCAL TEXT POINTER z1, z2;                                       059
      REF authorstr;  %address of string containing ident(s) of new
      author(s)%                                                       060
      %---------------------%                                         061
      astruc(&authorstr); %Set it upper case%                          062
      FIND jwp1 > ["Author(s):"] ['/] ↑z1 [';] ↑z2 ←z2;                063
      ST z1 z2 ← *authorstr*;                                          064
      RETURN;                                                          065
      END.
                                                                       066
   (setjaction) %set Action Distribution field in JWORK header%
   PROCEDURE (recipientstr);                                           067
      REF recipientstr;  %address c string containing list of "action"
      recipients%                                                      068
      LOCAL TEXT POINTER z1,z2;                                        069
      %---------------------%                                         070
      FIND jwp1 >                                                      071
         ("Action Distribution: "] < ['A] SP ↑z1 > [';] ↑z2            072
         / ["Author(s): "] [';] ↑z1 ↑z2);                              073
      ST z1 z2 ←  " Action Distribution: /", *recipientstr*, ';;       074
      RETURN END.
                                                                       075
   (setjaxcess) %set the access list field in JWORK header%
   PROCEDURE (accessstr);                                              076
      %----------%                                                    077
```

```
   LOCAL TEXT POINTER z1, z2;                                          078
   REF accessstr:                                                      079
   %----------%                                                        080
   FIND jwp1 > ([*AccessList: *] < ['A] CH ↑z1 > [';] ↑z2 /            081
      ([*Updates Document(s): *] /                                     082
      [*Obsoletes Document(s): *] /                                    083
      [*RFC# *] /                                                      084
      [*Sub-Collections: *] /                                          085
      [*Keywords: *] /                                                 086
      [*Distribution: *] /                                             087
      [*Author(s): *]) [';] ↑z1 ↑z2);                                  088
   ST z1 z2 ← " AccessList: ", *accessstr*, ';;                        089
   RETURN;                                                             090
   END.                                                                091
                                                                       092
(setiinfo) %set Information Distribution field in JWORK header%
PROCEDURE (recipientstr);                                              093
   REF recipientstr;  %address o string containing list of
   "information only" recipients%                                      094
   LOCAL TEXT POINTER z1,z2;                                           095
   %--------------------%                                             096
   FIND jwp1 >                                                         097
      ([*Information-only Distribution: *] < ['I] SP ↑z1 > [';] ↑z2
                                                                       098
      / [*Author(s): *] [';] ↑z1 ↑z2);                                 099
   ST z1 z2 ←  " Information-only Distribution: /", *recipientstr*,
   ';;                                                                 0100
   RETURN END.
                                                                       0101
(setjcomment) %set Comment field in JWORK header%
PROCEDURE (commentstr);                                                0102
   REF commentstr; %address of string containing desire comment%       0103
   LOCAL TEXT POINTER z1,z2;                                           0104
   %---------------%                                                  0105
   %remove " from the string%                                          0106
      FIND SF(*commentstr*);                                           0107
      LOOP CASE READC OF                                               0108
         = '"; %replace it with a single quote%                        0109
            BEGIN                                                      0110
            FIND ↑z2 < CH > ↑z1;                                       0111
            ST z1 z2 ← '';                                             0112
            FIND z2 >;                                                 0113
            END;                                                       0114
         = ENDCHR: EXIT LOOP;                                          0115
         ENDCASE ;                                                     0116
   FIND jwp1 > [*Clerk: *] [*
                                                                       0117
   *] ↑z1;
   ST jwp1 ← SF(jwp1) z1, ".PEL;  .PN=PN-1; .GCR;", *commentstr*;
                                                                       0118
   RETURN END.
                                                                       0119
(setjexpedite) %set or clear expedite attribute in JWORK header%
PROCEDURE (onoff);                                                     0120
   %IF onoff = TRUE, set expedite attribute, otherwise clear it%       0121
   LOCAL TEXT POINTER z1,z2;                                           0122
```

```
   %---------------%                                                    0123
   FIND jwp1 >                                                         0124
      (["(Expedite) "] ↑z2 < ['(] SP > ↑z1 /                           0125
      (["Title: "] < ['T] / ["Author(s): "] < ['A]) SP > ↑z1 ↑z2);
                                                                       0126
   IF onoff THEN ST z1 z2← " (Expedite) "                              0127
   ELSE ST z1 z2 ← NULL;                                               0128
   RETURN END.
                                                                       0129
(setjnumber) %set the contents of the Number field in the JWORK
header%
PROCEDURE (numberstr);                                                 0130
   REF numberstr;                                                      0131
   LOCAL TEXT POINTER z1,z2;                                           0132
   %---------------%                                                   0133
   FIND jwp1 > [";;;;"] ['(] ('J/'j) ↑z1 [')] ↑z2 ←z2;                 0134
   ST z1 z2 ← *numberstr*;                                             0135
   RETURN;                                                             0136
   END.
                                                                       0137
(setjkeyword) %set keywords field in JWORK header%
PROCEDURE (keywordstr);                                                0138
   REF keywordstr; %address of string containing list of keywords%
                                                                       0139
   LOCAL TEXT POINTER z1,z2;                                           0140
   %---------------%                                                   0141
   FIND jwp1 >                                                         0142
      (["Keywords: "] < ['K] SP ↑z1 > [';] ↑z2 /                       0143
      (["Distribution: "]/["Author(s): "]) [';] ↑z1 ↑z2);               0144
   ST z1 z2 ← " Keywords: ", *keywordstr*, ';;                         0145
   RETURN END.
                                                                       0146
(setjlink) %set INSERT LINK (***FIX THIS***) field in JWORK header%
PROCEDURE (keywordstr);                                                0147
   REF keywordstr; %address of string containing list of keywords%
                                                                       0148
   LOCAL TEXT POINTER z1,z2;                                           0149
   %---------------%                                                   0150
   FIND jwp1 >                                                         0151
      (["Keywords: "] < ['K] SP ↑z1 > [';] ↑z2 /                       0152
      (["Distribution: "]/["Author(s): "]) [';] ↑z1 ↑z2);               0153
   ST z1 z2 ← " Keywords: ", *keywordstr*, ';;                         0154
   RETURN END.
                                                                       0155
(setjrfc) %set RFC field in JWORK header%
PROCEDURE (rfcstr);                                                    0156
   REF rfcstr; %address of string containing RFC number%              0157
   LOCAL TEXT POINTER z1,z2;                                           0158
   %---------------%                                                   0159
   FIND jwp1 >                                                         0160
      (["RFC# "] < ['R] SP ↑z1 > [';] ↑z2 /                            0161
      (["Sub-Collections: "] / ["Keywords: "] / ["Distribution: "] /
      ["Author(s): "]) [';] ↑z1 ↑z2);                                   0162
   ST z1 z2 ← " RFC# ", *rfcstr*, ';;                                  0163
   RETURN END.
                                                                       0164
```

```
(setjsource) %set source type and content in JWORK header%
PROCEDURE (sourcetype, s1, s2);                                          0165
   % souce may be of type statement, group, file, or hardcopy %          0166
   LOCAL TEXT POINTER w1, w2, t1, t2;                                    0167
   LOCAL STRING locstr[200];                                             0168
   REF s1, s2;                                                           0169
   %------------------%                                                  0170
   w1[1] ← w2[1] ← 1;                                                    0171
   FIND jwp1 > (";;;; "/ ↑t1 CH ↑t2;                                     0172
   FIND t1;                                                              0173
   CASE READC OF                                                         0174
      = 'H:  % remove hardcopy location field %                          0175
         BEGIN                                                           0176
         IF FIND jwp1 > [')] ["Hard Copy--Location: "/ < ['H] SP ↑w1
         > [';] ↑w2 THEN                                                 0177
            ST w1 w2 ← NULL;                                             0178
         END;                                                            0179
      = 'F:  % remove origin field %                                     0180
         BEGIN                                                           0181
         IF FIND jwp1 > ["Clerk: "/ ["Origin: "/ < ['O] SP ↑w1 >
         ["####;"/ ↑w2 THEN                                              0182
            ST w1 w2 ← NULL;                                             0183
         END;                                                            0184
      = 'S: NULL;  % just replace substructure is  ok? %                 0185
      ENDCASE err(#"Illegal Journal Workfile format -- setjsource");
                                                                         0186
   IF ( w1 ← getsub(jwp1) ) NOT= jwp1 THEN                               0187
      BEGIN  % delete old source %                                       0188
      w2 ← getail(w1);                                                   0189
      cdelgro(w1, w2, FALSE, 0);                                         0190
      END;                                                               0191
   CASE sourcetype OF                                                    0192
      =stmtv:  %statement or text%                                       0193
         BEGIN                                                           0194
          % insert new source %                                          0195
            cinssta(jwp1, levdown, &s1, &s2);                            0196
         ST t1 t2 ← 'F;                                                  0197
         END;                                                            0198
      =groupv:  % group (branch, plex) %                                 0199
         BEGIN % insert new group as substructure of header %            0200
         IF s1.stastr THEN %user typed 'group' text%                     0201
            cinssta(jwp1, levdown, &s1, &s2)                             0202
         ELSE % normal group %                                           0203
            ccopgrc(jwp1, levdown, s1, s2, FALSE, 0);                    0204
         ST t1 t2 ← 'F;                                                  0205
         END;                                                            0206
      =filev: % whole file %                                             0207
         BEGIN                                                           0208
         %load the file%                                                 0209
            caddexp(&s1, &s2, lda(), &w1);                               0210
            IF w1 = endfil THEN                                          0211
               BEGIN                                                     0212
               *locstr* ← "Illegal link: ", s1 s2;                       0213
               err(&locstr);                                             0214
               END;                                                      0215
            s1 ← orgstid;  s1.stfile ← w1.stfile;                        0216
```

```
        %copy origin statement into jwork header%                    0217
           IF NOT FIND jwp1 ["Clerk:"] [EOL EOL] ↑w1 ←w1 ←w1 THEN
                                                                     0218
              err($"Improper Journal Workfile Format --
              setjsource");                                          0219
           ST w1 w1 ← " Origin: ", SF(s1) SE(s1), "####;";           0220
        %copy substructure to jwork file%                            0221
           IF (s1 := getsub(s1)) # s1 THEN % there is substructure
           in user's file %                                          0222
              BEGIN                                                  0223
              s2 ← getail(s1);                                       0224
              REPEAT CASE (Groupv);                                  0225
              END;                                                   0226
           % no substructure in user's file -- boy is he stupid! %
                                                                     0227
              *locstr* ← "[no content was specified!]";              0228
              FIND SF(*locstr*) ↑s1 SE(*locstr*) ↑s2;                0229
              REPEAT CASE (stmtv);                                   0230
        END;                                                         0231
     =forwdv:  % forward an item to specified recipients %           0232
        BEGIN                                                        0233
         % insert forward request %                                  0234
           cinssta(jwp1, levdown, $s1, $s2);                         0235
        ST t1 t2 ← 'S;                                               0236
        END;                                                         0237
     =hcopyv:  % hardcopy item %                                     0238
        BEGIN                                                        0239
        FIND t2 [')] ↑w1;                                            0240
        ST t1 ← SF(t1) t1, 'H, t2 w1, " Hard Copy--Location: ", s1
        s2, ';, w1 SE(w1);                                           0241
        END;                                                         0242
     ENDCASE err($"Illegal source type in setjsource");              0243
  RETURN;                                                            0244
  END.                                                               0245
(setjsubcol) %set Sub-Collections field in JWORK header%
PROCEDURE (subcollstr);                                              0246
  LOCAL STRING tempsr[150];                                          0247
  REF subcollstr; %address of string containing list of
  Sub-Collections%                                                   0248
  LOCAL TEXT POINTER z1,z2;                                          0249
  %---------------%                                                 0250
  *tempsr* ← *subcollstr*;                                           0251
  astruc($tempsr); %Make sure that it is upper case%                 0252
  FIND SF(jwp1)                                                      0253
     (["Sub-Collections: "] < ['S] SP ↑z1 > [';] ↑z2 /               0254
     ( ["Keywords: "] / ["Distribution: "] / ["Author(s): "]) [';]
     ↑z1 ↑z2);                                                       0255
  ST z1 z2 ← " Sub-Collections: ", *tempsr*, ';;                     0256
  RETURN END.
                                                                     0257
(setjobsoletes) %set Obsoletes Document(s) field in JWORK header%
PROCEDURE (obsoletestr);                                             0258
  LOCAL TEXT POINTER z1, z2;                                         0259
  REF obsoletestr; %address of string containing list of Obsoleted
  Documents%                                                         0260
  %---------------%                                                 0261
```

```
   FIND jwp1 >                                                          0262
      (["Obsoletes Document(s): "] < ['O] SP ↑z1 > [';] ↑z2/            0263
      (["RFC# "] / ["Sub-Collections: "] / ["Keywords: "] /
      ["Distribution: "] / ["Author(s): "]) [';] ↑z1 ↑z2);               0264
   ST z1 z2 ← " Obsoletes Document(s): ", *obsoletestr*, '; ;           0265
   RETURN;                                                              0266
   END.
                                                                        0267
(set.unrecorded) %set or clear UNRECORDED attribute in JWORK heade %
PROCEDURE (onoff);                                                      0268
   %IF onoff = TRUE, set unrecorded attribute, otherwise clear it%
                                                                        0269
   LOCAL TEXT POINTER z1,z2;                                            0270
   %---------------%                                                   0271
   FIND jwp1 >                                                          0272
      (["(Unrecorded) "] ↑z2 < ['(] SP > ↑z1 /                          0273
      (["Title: "] < ['T] / ["Author(s): "] < ['A]) SP > ↑z1 ↑z2);
                                                                        0274
   IF onoff THEN ST z1 z2← " (Unrecorded) "                             0275
   ELSE ST z1 z2 ← NULL;                                                0276
   RETURN END.
                                                                        0277
(set.updates) %set Updates Document(s) field in JWORK neader%
PROCEDURE (updatestr);                                                  0278
   LOCAL TEXT POINTER z1, z2;                                           0279
   REF updatestr; %address of string containing list of Updated
   Documents%                                                           0280
   %---------------%                                                   0281
   FIND jwp1 >                                                          0282
      (["Updates Document(s): "] < ['U] SP ↑z1 > [';] ↑z2/              0283
      (["Obsoletes Document(s): "] / ["RFC# "] / ["Sub-Collections:
      "] / ["Keywords: "] / ["Distribution: "] / ["Author(s): "])
      [';] ↑z1 ↑z2);                                                    0284
   ST z1 z2 ← " Updates Document(s): ", *updatestr*, '; ;               0285
   RETURN;                                                              0286
   END.
                                                                        0287
(set.title)%set Title field in JWORK header%
PROCEDURE(titlestr);                                                    0288
   REF titlestr; %address of string containing desired title%           0289
   LOCAL TEXT POINTER z1,z2;                                            0290
   %---------------%                                                   0291
   %remove " from the string%                                           0292
      FIND SF(*titlestr*);                                              0293
      LOOP CASE READC OF                                                0294
         = '": %replace it with a single quote%                         0295
            BEGIN                                                       0296
            FIND ↑z2 < CH > ↑z1;                                        0297
            ST z1 z2 ← '';                                              0298
            FIND z2 >;                                                  0299
            END;                                                        0300
         = ENDCHR: EXt LOOP;                                           0301
         ENDCASE ;                                                      0302
   FIND jwp1 >                                                          0303
      (["Title: .H1="]  < ['T] SP ↑z1 > [';] ↑z2/                       0304
       ["Author(s): "] < ['A] SP ↑z1 ↑z2>);                             0305
```

```
    ST z1 z2 ← " Title: .R1=", '",  *titlestr*, '", ';;                    0306
    RETURN END.
                                                                           0307
(setcacc) %set access field in catalog entry%
PROCEDURE (entrystid, repstr, ptr1, ptr2);                                 0308
    %----------%                                                          0309
    LOCAL repcnt;                                                          0310
    LOCAL TEXT POINTER z1, z2;                                             0311
    REF repstr, ptr1, ptr2;                                                0312
    %----------%                                                          0313
    IF NOT &ptr1 THEN getcacc (entrystid, 0, $z1, $z2)                     0314
    ELSE FIND ptr1 ↑z1 ptr2 ↑z2;                                           0315
    IF *repstr* = *nullfield* THEN %delete the field%                      0316
       IF (FIND z1 < [$PT] $NP > ↑z1 ["AccessList:"] z2 [';] ↑z2)
       THEN                                                                0317
          ST entrystid ← SF(entrystid) z1, z2 SE(entrystid)                0318
       ELSE NULL                                                           0319
    ELSE                                                                   0320
       BEGIN                                                               0321
       repcnt ← repstr.L;                                                  0322
       IF *repstr* [repstr.L] = '; THEN BUMP DOWN repcnt;                  0323
       IF NOT (FIND z1 < [$PT] $NP > ["AccessList:"]) THEN                 0324
          ST entrystid ← SF(entrystid) z1, " AccessList: ", *repstr*
          [1 TO repcnt], ';, z2 SE(entrystid)                              0325
       ELSE                                                                0326
          ST entrystid ← SF(entrystid) z1, *repstr* [1 TO repcnt], z2
          SE(entrystid);                                                   0327
       END;                                                                0328
    RETURN;                                                                0329
    END.                                                                   0330
                                                                           0331
%....Status....%                                                           0332
    (jstatus) %generate a submission status string%
    PROCEDURE (status);                                                    0333
       % jstatus documentation                                            0334
          It works by calling a set of getj routines, which are roughly
          complementary to the setj routines including the fact that
          they expect jwpl to be set to the first character of the JWORK
          origin statement.                                                0335
          Each getj routine either null's the passed string if the field
          does not exist or fills it with the value of the field.          0336
          The results are simply appended and separated by EOL's to form
          a longer string which is printed out to the user.                0337
          %                                                                0338
       LOCAL stid;                                                         0339
       LOCAL TEXT POINTER z1, z2;                                         01260
       LOCAL STRING str[1500];                                             0340
       REF status;  %address of string into which the status message is
       to be stored%                                                       0341
       %--------------------%                                             0342
       *status*  ←                                                         0343
          *[addeol ( getjnumber ( $str ), $sjnumber )]*,                   0344
          *[addeol ( getjaxcess ( $str ), $sjaccess )]*,                   0345
          *[addeol ( getjtitle ( $str ), $sjtitle )]*,                     0346
          *[addeol ( getjcomment ( $str ), $sjcomment )]*,                 0347
          *[addeol ( getjauthor ( $str ), $sjauthor )]*,                   0348
```

```
    *[addeol ( getjaction ( $str ), $sjaction )]*,                    0349
    *[addeol ( getjinfo ( $str ), $sjinfo )]*,                        0350
    *[addeol ( getjsubcol ( $str,0 ), $sjsubcol  )]*;                 0351
*status* ← *status*, %this is broken to avoid an L1O STACK
OVERFLOW%                                                             0352
    *[addeol ( getjkeyw ( $str ), $sjkeywords )]*,                    0353
    *[addeol ( getjexpedite ($str), $sjhandling)]*,                   0354
    *[addeol ( getjunrecorded ($str), $sjrecording)]*,                0355
    *[addeol ( getjrfc ( $str ), $sjrfc )]*,                          0357
    *[addeol ( getjobsoletes ( $str ), $sjobsoletes )]*,              0358
    *[addeol ( getjupdates ( $str ), $sjupdates )]*,                  0359
    *[addeol ( getjlink ( $str ), $sjlink )]*;                        0360
getjtype( $str );                                                     0361
CASE *str*[1] OF  %type of source%                                    0362
    = 'M, = 'F:                                                      01282
      BEGIN  %determine what status to give%                         01257
      % Is it a file?%                                               01261
      IF FIND SF(jwpl) ["Clerk: "] ["Origin: "] ↑z1 [';]['';] ↑z2
      THEN                                                           01259
         *status* ← *status*, EOL, *sjfile*, z1 z2, EOL              01262
      ELSE  %see if it is a structure%                               01263
         IF (stid ← getnxt(jwpl)) NOT= endfil THEN % There is
         substructure%                                               01254
            IF getnxt(stid) = endfil THEN                            01255
               IF NOT FIND SF(stid) *nullsource* ENDCHR THEN         01264
                  *status* ← *status*, EOL, *sjmessage*, SF(stid)
                  SE(stid), EOL                                      0363
               ELSE NULL  %user has not supplied source yet%         01265
            ELSE %show some text from first and last branches%
                                                                     01256
               BEGIN                                                 01271
               FIND SF(stid) ↑z1 $15CH ↑z2;                          01272
               IF getftl(stid) THEN % a branch%                      01268
                  *status* ← *status*, EOL, "BRANCH BEGINING: ",
                  '", z1 z2, '", EOL                                 01270
               ELSE                                                  01269
                  BEGIN                                              01275
                  *status* ← *status*, EOL, "GROUP BEGINING: ",
                  '", z1 z2, '", EOL;                                01274
                  stid ← getail(stid);                               01278
                  FIND SF(stid) ↑z1 $15CH ↑z2;                       01276
                  *status* ← *status*, "  [THROUGH]  ", '", z1
                  z2, '", EOL;                                       01279
                  END;                                               01277
               END;                                                  01273
      END;                                                           01258
    = 'H:  %hardcopy%                                                01284
      *status* ← *status*,                                           01286
         *[addeol ( getjhcloc ( $str ), $sjhardcopy )]*;              0356
    = 'S: %secondary distribution%                                   01285
      IF (stid ← getnxt(jwpl)) NOT= endfil THEN % There is
      substructure%                                                  01288
         *status* ← *status*, EOL, *sjforward*, SF(stid)
         SE(stid), EOL;                                              01287
    ENDCASE;                                                         01283
RETURN (&status);                                                     0364
```

```
    END.                                                                0365

    (addeol) %support routine for jstatus%
    PROCEDURE (string, heading);                                        0366
       %*addeol*                                                        0367
          This routine is used by jstatus for inserting EOL's between
          fields.                                                       0368
          Basically, if it is handed a NULL string, then it does
          nothing, but if handed a non-empty string, it appends an EOL.
                                                                        0369
          %                                                             0370
       REF string, heading;                                             0371
       %-----------------%                                              0372
       IF string.L > 0 THEN *string* ← *heading*, SP, *string*, EOL;    0373
       RETURN(&string);                                                 0374
       END.
                                                                        0375
%....Get contents of Journal header fields....%                         0376
    %                                                                   0377
    This is a set of routines which return values of fields in the
    Journal header .                                                    0378
    Generally, they assume jwpl is a pointer to the first character of
    the statement containing the header, and search that statement for
    the presence of the field.                                          0379
    If the field is not found, the string address is set to  NULL.      0380
    Otherwise, the contents of the field is copied into the passed
    string, and its address is returned.                                0381
    In cases where the getj routines do unusual things, they are
    documented separately                                               0382
    %                                                                   0383
    (getjaction) %get the contents of the Action Distribution field from
    the JWORK header%
    PROCEDURE (string);                                                 0384
       REF string;  LOCAL TEXT POINTER z1, z2;                          0385
       IF FIND jwpl > ["Action Distribution: "] ['//] ↑z1 [';] ↑z2 ←z2
       THEN                                                             0386
          *string* ← z1 z2                                              0387
       ELSE *string* ← NULL;                                            0388
       RETURN(&string);                                                 0389
       END.
                                                                        0390
    (getjauthor) %get the contents of the Author field from the JWORK
    header%
    PROCEDURE (string);                                                 0391
       REF string;  LOCAL TEXT POINTER z1, z2;                          0392
       FIND jwpl > ["Author(s):"] ['//] ↑z1 [';] ↑z2 ←z2;               0393
       *string* ← z1 z2;                                                0394
       RETURN(&string);                                                 0395
       END.
                                                                        0396
    (getjaxcess) %get the contents of the Access List field from the
    JWORK header%
    PROCEDURE (string);                                                 0397
       REF string;  LOCAL TEXT POINTER z1, z2;                          0398
       IF rdprvsts (jwpl.stfile) = spsprivate THEN                      0399
          *string* ← "PRIVATE"                                          0400
```

```
   ELSE *string* ← NULL;                                                0401
   RETURN(&string);                                                     0402
   END.

                                                                        0403
(getjclerk) %get the contents of the Clerk field from the JWORK
header%
PROCEDURE (string);                                                     0404
   REF string;  LOCAL TEXT POINTER z1, z2;                              0405
   FIND jwp1 > /"Clerk: "/ ↑z1 /'.]/ ↑z2 ←z2;                           0406
   *string* ← z1 z2;                                                    0407
   RETURN(&string);                                                     0408
   END.
                                                                        0409
(getjcomment) %get the contents of the Comment field from the JWORK
header%
PROCEDURE (string);                                                     0410
   REF string;  LOCAL TEXT POINTER z1, z2;                              0411
   *string* ← NULL;                                                     0412
   IF ( FIND jwp1 > /"

   "/ /".GCR;"/  ↑z1 ) AND ( POS z1 # SE(z1) ) THEN                     0413
      *string* ← z1 SE(z1);                                             0414
   RETURN(&string);                                                     0415
   END.
                                                                        0416
(getjexpedite) %get the Expedite attribute from the #jwork header%
PROCEDURE (string);                                                     0417
   REF string;  LOCAL TEXT POINTER z1, z2;                              0418
   *string* ← NULL;                                                     0419
   IF FIND jwp1 > /"(Expedite)"/ THEN *string* ← "Expedite";            0420
   RETURN(&string);                                                     0421
   END.
                                                                        0422
(getjunrecorded) %get the Unrecorded attribute from the JWORK
header%
PROCEDURE (string);                                                     0423
   REF string;  LOCAL TEXT POINTER z1, z2;                              0424
   *string* ← #null:
   if find jwp1 > /"(Unrecorded)"/ THEN *string* ← "Unrecorded";        0426
   RETURN(&string);                                                     0427
   END.
                                                                        0428
(getjhcloc) %get the contents of the Hard Copy Location field from
the JWORK header%
PROCEDURE (string);                                                     0429
   REF string;  LOCAL TEXT POINTER z1, z2;                              0430
   *string* ← NULL;                                                     0431
   IF FIND jwp1 > /"Hard Copy--Location: "/ ↑z1 /';/ ↑z2 ←z2 THEN
                                                                        0432
      *string* ← z1 z2;                                                 0433
   RETURN(&string);                                                     0434
   END.
                                                                        0435
(getjinfo) %get the contents of the Information-only Distribution
field from the JWORK header%
```

```
PROCEDURE (string);                                                    0436
   REF string;  LOCAL TEXT POINTER z1, z2;                             0437
   IF FIND jwp1 > [("Information-only Distribution: "] [('/] ↑z1 [';]
   ↑z2 ←z2 THEN                                                        0438
      *string* ← z1 z2                                                 0439
   ELSE *string* ← NULL;                                               0440
   RETURN(&string);                                                    0441
   END.
                                                                       0442
(get.jlink) %get the contents of the Insert Link field from the JWORK
header%
PROCEDURE (string);                                                    0443
   REF string;  LOCAL TEXT POINTER z1, z2;                             0444
   IF FIND jwp1 > ["Insert Link at: "] ↑z1 [';] ↑z2 ←z2 THEN           0445
      *string* ← z1 z2                                                 0446
   ELSE *string* ← NULL;                                               0447
   RETURN(&string);                                                    0448
   END.
                                                                       0449
(get.jkeyw) %get the contents of the Keywords field from the JWORK
header%
PROCEDURE (string);                                                    0450
   REF string;  LOCAL TEXT POINTER z1, z2;                             0451
   IF FIND jwp1 > ["Keywords: "] ↑z1 [';] ↑z2 ←z2 THEN                 0452
      *string* ← z1 z2                                                 0453
   ELSE *string* ← NULL;                                               0454
   RETURN(&string);                                                    0455
   END.

                                                                       0456
(get.jnumber) %get the contents of the Number field from the JWORK
header%
PROCEDURE (string);                                                    0457
   REF string;  LOCAL TEXT POINTER z1, z2;                             0458
   IF NOT FIND jwp1 > [";;;;"]['(] ('J/'j) ↑z1 $D ↑z2 THEN             0459
      err($"Bad Journal Work File");                                   0460
   *string* ← z1 z2;                                                   0461
   RETURN(&string);                                                    0462
   END.
                                                                       0463
(get.jtype) %get the TYPE of the item from the JWORK header%
PROCEDURE (string);                                                    0464
   REF string;  LOCAL TEXT POINTER z1, z2;                             0465
   IF NOT FIND jwp1 > [";;;; "] ↑z1 CH ↑z2 THEN                        0466
      err($"Bad Journal Work File");                                   0467
   *string* ← z1 z2;                                                   0468
   RETURN(&string);                                                    0469
   END.
                                                                       0470
(get.jobsoletes) %get the contents of the Obsoletes Documents field
from the JWORK header%
PROCEDURE (string);                                                    0471
   REF string;  LOCAL TEXT POINTER z1, z2;                             0472
   IF FIND jwp1 > ["0 BSOLETES Document(s): "]  ↑z1 [';] ↑z2 ←z2 THEN
                                                                       0473
      *string* ← z1 z2                                                 0474
```

```
   ELSE *string* ← NULL;                                                0475
   RETURN(&string);                                                     0476
   END.
                                                                        0477
(getjrfc) %get the contents of the RFC field from the JWORK header%
PROCEDURE (string);                                                     0478
   REF string;  LOCAL TEXT POINTER z1, z2;                              0479
   IF FIND jwp1 > /"RFC# "/   ↑z1 /';/ ↑z2 ←z2 THEN                     0480
      *string* ← z1 z2                                                  0481
   ELSE *string* ← NULL;                                                0482
   RETURN(&string);                                                     0483
   END.
                                                                        0484
(getjsubcol) %get the contents of the Sub-Collections field from the
JWORK header%
PROCEDURE (string);                                                     0485
   REF string;  LOCAL TEXT POINTER z1, z2;                              0486
   IF (FIND jwp1 > /"Sub-Collections: "/   ↑z1 /';/ ↑z2 ←z2 ) THEN
                                                                        0487
      *string* ← z1 z2                                                  0488
   ELSE *string* ← NULL;                                                0489
   RETURN(&string);                                                     0490
   END.
                                                                        0491
(galjsubcol)PROCEDURE(string, idfnum);                                  01203
   %*galjsubcol-- get all jsubcollection *                              01204
      This is the getj routine which returns the subcollections
      which the document belongs to.                                    01205
      It is a little more complex than most getj routines, because
      the subcollection membership is computed from more than one
      field, and redundant entries must be deleted.                     01206
      The algorithm is:                                                 01207
         Get value of sub-collection field in header                    01208
         If this is an RFC document, add NWG and NIC to
         subcollections                                                 01209
         If there are any groups on the distribution list for this
         document, enter them into te subcollection list (this uses
         the routine getgpids).                                         01210
      As usual, idfile is the file number of the identfile or zero.
                                                                        01211
      %                                                                 01212
   LOCAL TEXT POINTER z1, z2, z3;                                       01213
   LOCAL STRING tempsr[150], templsr[30];                               01214
   REF string;                                                          01215
   IF (FIND jwp1 > /"Sub-Collections: "/   ↑z1 /';/ ↑z2 ←z2 ) THEN
                                                                        01217
      *string* ← z1 z2                                                  01218
   ELSE *string* ← NULL;                                                01238
   IF rfcflg THEN *string* ← "NWG ", *string*;                          01219
   IF rfcflg AND NOT FIND BETWEEN z1 z2(/"NIC"/) THEN                   01220
      *string* ← "NIC ", *string*;                                      01221
   IF FIND jwp1 > /"Distribution: "/ /'//  ↑z1 THEN                     01222
      BEGIN                                                             01223
         *tempsr* ← NULL;                                               01224
         getgpids(*z1, $tempsr, idfnum); %Extract all group
         references from ident list%                                    01225
```

```
            %Now fix up Sub-collection Field to reflect Distribution
            Groups%                                                    01226
               FIND SF(*tempsr*) ↑z1;                                  01227
               WHILE (FIND z1 $NP ↑z2 1SPT ↑z1) DO                     01228
                  BEGIN                                                01229
                  *templsr* ← +z2 z1;                                  01230
                  IF NOT FIND SF(*string*) [*templsr*] THEN            01231
                     *string* ← *string*, SP, *templsr*;               01232
                  END;                                                 01233
            END;                                                       01234
         RETURN(&string);                                              01236
         END.
                                                                       01237
      (getjtitle) %get the contents of the Title field from the JWORK
      header%
      PROCEDURE (string);                                              0492
         REF string;  LOCAL TEXT POINTER z1, z2;                       0493
         IF FIND jwp1 > ["Title: "] ['"] -'" ↑z1 ←z1 ['"] ↑z2 ←z2 THEN 0494
            *string* ← z1 z2                                           0495
         ELSE *string* ← NULL;                                         0496
         RETURN(&string);                                              0497
         END.
                                                                       0498
      (getjupdates) %get the contents of the Updates Documents field from
      the JWORK header%
      PROCEDURE (string);                                              0499
         REF string;  LOCAL TEXT POINTER z1, z2;                       0500
         IF FIND jwp1 > ["Updates Document(s): "] ↑z1 [';] ↑z2 ←z2 THEN
                                                                       0501
            *string* ← z1 z2                                           0502
         ELSE *string* ← NULL;                                         0503
         RETURN(&string);                                              0504
         END.
                                                                       0505
   %....catalog manipulation routines....%                             0506
      (getcacc) %fetch access field from catalog entry%
      PROCEDURE (entrystid, fldstr, ptr1, ptr2);                       0507
         %----------%                                                  0508
         LOCAL retval;                                                 0509
         LOCAL TEXT POINTER z1, z2, z3;                                0510
         %----------%                                                  0511
         retval ← FALSE;                                               0512
         makeptr (entrystid, $z3);                                     0513
         IF NOT fndunqtxt ($"AccessList:", $z3, $z1, $z2)              0514
         OR NOT (FIND z2 > $NP ↑z1 [';] < CH $NP ↑z2) THEN             0515
            getcend (entrystid, $z1, $z2)                              0516
         ELSE retval ← TRUE;                                           0517
         stptset (fldstr, ptr1, ptr2, $z1, $z2);                       0518
         RETURN (retval);                                              0519
         END.                                                          0520
                                                                       0521
      (getcfn) %fetch filename from catalog entry%
      PROCEDURE (entrystid, fldstr, ptr1, ptr2);                       0522
         %----------%                                                  0523
         LOCAL linkstid, retval;                                       0524
         LOCAL TEXT POINTER tp1, tp2;                                  0525
```

```
    REF fldstr;                                                          0526
    %----------%                                                         0527
    retval ← FALSE;                                                      0528
    IF NOT &fldstr THEN err ();                                          0529
    linkstid ← getsub (entrystid);                                       0530
    IF NOT (FIND SF(linkstid) ["Link"] ['(/ ↑tp1 ←tp1 [')] ↑tp2) THEN
                                                                         0531
       getcend (entrystid, $tp1, $tp2)                                   0532
    ELSE                                                                 0533
       BEGIN                                                             0534
       lnbfls ($tp1, 0, &fldstr);                                        0535
       *fldstr* ← *fldstr*, ".NLS";                                      0536
       retval ← TRUE;                                                    0537
       END;                                                              0538
    stptset (0, ptr1, ptr2, $tp1, $tp2);                                0539
    RETURN (retval);                                                     0540
    END.                                                                 0541
                                                                         0542
(getcend) %fetch end of catalog entry%
PROCEDURE (entrystid, ptr1, ptr2);                                       0543
    %----------%                                                         0544
    REF ptr1, ptr2;                                                      0545
    %----------%                                                         0546
    FIND SE(entrystid) ↑ptr1 ↑ptr2;                                      0547
    RETURN;                                                              0548
    END.
                                                                         0549
(gtcatent) %locate document in catalogs%
PROCEDURE (docnumber, %opt% catfilename, %or% catfileno,
openformodify);                                                          0550
    %----------%                                                         0551
    LOCAL stid, catalog, fileno, routine, curskpachk;                    0552
    LOCAL STRING catstmtname [10];                                       0553
    LOCAL TEXT POINTER z1;                                               0554
    REF docnumber, catfilename;                                          0555
    %----------%                                                         0556
    IF catfileno THEN %search for statement in open catalog file%        0557
       BEGIN                                                             0558
       %build text pointer to origin statement%                          0559
          stid ← 0;                                                      0560
          stid.stpsid ← origin;                                          0561
          stid.stfile ← catfileno;                                       0562
          makeptr (stid, $z1);                                           0563
       *catstmtname* ← 'J, *docnumber*;                                  0564
       lookup ($z1, $catstmtname, nametyp);                              0565
       RETURN (IF z1 # endfil1 THEN z1 ELSE FALSE);                      0566
       END;                                                              0567
    IF &catfilename THEN %open and search catalog file%                  0568
       BEGIN                                                             0569
       IF openformodify THEN                                             0570
          BEGIN                                                          0571
          routine ← $openlock;                                           0572
          conjdir (TRUE);                                                0573
          END                                                            0574
       ELSE routine ← $open;                                             0575
       fileno ← 0;                                                       0576
```

```
        curskpachk ← skpachk;                                       0577
        ON SIGNAL ELSE                                              0578
           BEGIN                                                    0579
           skpachk ← curskpachk;                                    0580
           IF fileno THEN sigclose (fileno := 0);                   0581
           IF openformodify THEN conjdir (openformodify ← FALSE);   0582
           END;                                                     0583
        skpachk ← TRUE;                                             0584
        fileno ← /routine/ (0, jflname (&catfilename));             0585
        IF openformodify THEN fileno ← fileno.stfile;               0586
        IF NOT (stid ← gtcatent (&docnumber, 0, fileno,
        openformodify)) THEN                                        0587
           BEGIN                                                    0588
           close (fileno := 0);                                     0589
           IF openformodify THEN conjdir (openformodify ← FALSE);   0590
           END;                                                     0591
        skpachk ← curskpachk;                                       0592
        RETURN (stid);                                              0593
        END;                                                        0594
     %try all catalog files%                                        0595
        IF (stid ← gtcatent (&docnumber, $"Tjcat", 0, openformodify))
                                                                    0596
        OR (stid ← gtcatent (&docnumber, $"Jcat", 0, openformodify))
        THEN                                                        0597
           RETURN (stid);                                           0598
        CASE srval (&docnumber, 10) OF                              0599
           < 12000: catalog ← $"Jcat0";                             0600
           < 16000: catalog ← $"Jcat1";                             0601
           < 20000: catalog ← $"Jcat2";                             0602
           < 24000: catalog ← $"Jcat3";                            01307
           ENDCASE RETURN (FALSE);                                  0603
        RETURN (gtcatent (&docnumber, catalog, 0, openformodify)); 0604
     END.                                                           0605
                                                                    0606
%....field conversion routines....%                                 0607
  (convjsubcol) %convert the contents of the Sub-Collections, Author,
  and Distribution field into a real Sub-Collections field for the
  JWORK header%
  PROCEDURE (idfile);                                               0608
     % convjsubcol documentation                                    0609
        This routine updates the sub-collections field for this
        document.                                                   0610
        The sub-collections membership is computed from the current
        sub-collection, author, and Distribution fields, and redundant
        entries are deleted.  If this is an RFC document, add NWG and
        NIC to subcollections.  If there are any groups on the
        distribution list for this document, enter them into the
        sub-collection list (this uses the routine getgpids).       0611
        %                                                           0612
     LOCAL rfcf;                                                    0613
     LOCAL TEXT POINTER z1, z2, a1, a2;                             0614
     LOCAL STRING                                                   0615
        string[300], tempsr[300], temp1sr[50], temp2sr[50],
        identry[1000];                                              0616
     %----------------%                                             0617
     *string* ← NULL;                                               0618
```

```
   IF (FIND jwpl > ["Sub-Collections: "]   ↑z1 [';] ) THEN          0619
       getgpids ($z1, $string, idfile);                             0620
   IF rfcf ← ( FIND jwpl > ["RFC # "] ) THEN                        0621
      IF NOT FIND SF(*string*) ["NWG"] THEN                         0622
         *string* ← *string*, " NWG";                               0623
   IF rfcf AND NOT FIND BETWEEN z1 z2(["NIC"]) THEN                 0624
      IF NOT FIND SF(*string*) ["NIC"] THEN                         0625
         *string* ← *string*, " NIC";                               0626
   getjauthor($tempsr);                                             0627
   %Now fix up Sub-collection Field to reflect authors%             0628
      FIND SF(*tempsr*) ↑z1;                                        0629
      WHILE (FIND z1 $NP ↑z2 1$PT ↑z1) DO                           0630
         BEGIN                                                      0631
         *templsr* ← +z2 z1; %an author%                            0632
         ckident($templsr, $identry, idfile); %get his ident entry%
                                                                    0633
         lgetsubcoll($identry, $temp2sr, 0, 0); %get his
         subcollections%                                            0634
         %process each element of his sub-coll list%                0635
            FIND SF(*temp2sr*) ↑a1;                                 0636
            WHILE (FIND a1 $NP ↑a2 1$PT ↑a1) DO                     0637
               BEGIN                                                0638
               *templsr* ← +a2 a1; %a sub-coll%                     0639
               IF NOT FIND SF(*string*) [*templsr*] THEN            0640
                  *string* ← *string*, SP, *templsr*;               0641
               END;                                                 0642
         END;                                                       0643
   IF FIND jwpl > ["Distribution: "] ['/] ↑z1 THEN  %FIX THIS LATER
   FOR ACTION, INFO DIST LIST%                                      0644
      BEGIN                                                         0645
         *tempsr* ← NULL;                                           0646
         getgpids($z1, $tempsr, idfile); %Extract all group
         references from ident list%                                0647
         %Now fix up Sub-collection Field to reflect Distribution
         Groups%                                                    0648
            FIND SF(*tempsr*) ↑z1;                                  0649
            WHILE (FIND z1 $NP ↑z2 1$PT ↑z1) DO                     0650
               BEGIN                                                0651
               *templsr* ← +z2 z1;                                  0652
               IF NOT FIND SF(*string*) [*templsr*] THEN            0653
                  *string* ← *string*, SP, *templsr*;               0654
               END;                                                 0655
      END;                                                          0656
   IF string.L > 0 THEN setjsubcol($string);                        0657
   RETURN;                                                          0658
   END.
                                                                    0659
(convjdist) %convert Action Distribution and Information-only
Distribution fields into a single Distribution field with comments
appended to recipient idents.%
PROCEDURE;                                                          0660
   LOCAL TEXT POINTER z1,z2, z3, z4;                                0661
   LOCAL STRING dist[500], idstr[25], comm[300], temp[400];         0662
   IF FIND jwpl > ["Action Distribution: "] ↑z2 < ['A] CH ↑z1 > z2
   ['/] $NP ↑z2 [';] ↑z4 < CH $NP > ↑z3 THEN %process action list%
                                                                    0663
```

```
        BEGIN %pointers z1 through z4 should be set up as in the
        following example:
           <z1> Action Distribution: / <z2>CHI DIA DCW<z3> ;<z4>  %
                                                                    0664
        *temp* ← z2 z3;  %membership%                               0665
        ST z1 z4 ← NULL;  %delete action list%                      0666
        %add ( / ACTION / ) to each ident; must agree with format
        checked in (nls, jnldel, distdl)%                           0667
           FIND SF(*temp*) ↑z1;                                     0668
           WHILE (FIND z1 $NP $(',) $NP ↑z2 1$(LD / '↑ / '& / '-) ↑z1)
           DO                                                       0669
              BEGIN                                                 0670
              *idstr* ← +z2 z1;                                     0671
              IF FIND z1 $NP '( ↑z2 ([')] ↑z1 ↑z3 ←z3 /[ENDCHR] ↑z1
              ↑z3) THEN                                             0672
                 *comm* ← z2 z3                                     0673
              ELSE *comm* ← NULL;                                   0674
              *dist* ← *dist*, *idstr*, '(, " [ ACTION ] ", *comm*,
              '), SP;                                               0675
              END;                                                  0676
           END;                                                     0677
        IF FIND jwp1 > ["Information-only Distribution: "] ↑z2 < ['I] CH
        ↑z1 > z2 ['/] $NP ↑z2 [';] ↑z4 < CH $NP > ↑z3 THEN %process
        information-only list%                                      0678
           BEGIN %pointers z1 through z4 should be set up as in the
           following example:
              <z1> Information-only Distribution: / <z2>RWW<z3> ;<z4>  %
                                                                    0679
           *temp* ← z2 z3;  %membership%                            0680
           ST z1 z4 ← NULL;  %delete info list%                     0681
           %add ( / INFO-ONLY / ) to each ident%                    0682
              FIND SF(*temp*) ↑z1;                                  0683
              WHILE (FIND z1 $NP $(',) $NP ↑z2 1$(LD / '↑ / '& / '-) ↑z1)
              DO                                                    0684
                 BEGIN                                              0685
                 *idstr* ← +z2 z1;                                  0686
                 IF FIND z1 $NP '( ↑z2 ([')] ↑z1 ↑z3 ←z3 /[ENDCHR] ↑z1
                 ↑z3) THEN                                          0687
                    *comm* ← z2 z3                                  0688
                 ELSE *comm* ← NULL;                                0689
                 *dist* ← *dist*, *idstr*, '(, " [ INFO-ONLY ] ", *comm*,
                 '), SP;                                            0690
                 END;                                               0691
           END;                                                     0692
        IF dist.L THEN                                              0693
           BEGIN                                                    0694
           FIND jwp1 > [" Author(s): "] [';] ↑z1 ↑z2;               0695
           ST z1 z2 ← " Distribution: /", *dist*, ';;               0696
           END;                                                     0697
        RETURN END.
                                                                    0698
    %Procedures concerned with locking/unlocking Journal%           0720
       (jolock)PROC;                                                0721
          %This procedure checks the global flags which tell the current
          status of te Journal:                                     0722
             Flag 0: (Password jlock): If on Return True, Otherwise False.
```

```
                                                                    0723
        This flag is used to prevent new users fromm entering the
        JOurnal, but does not stop persons who are already in the
        process of using it.                                        0724
      Flag 1: (password jbfil): If on, a file error has been noted
      in one of the JOurnal files.  Discontinnue the current Journal
      Process immediately with a Werr.                              0725
   %                                                                0726
   IF jdebug THEN RETURN(FALSE);                                    0727
   IF flagut(1, $tstfg) THEN                                        0728
      %Some one has found a bad file somewhere%                     0729
      werr($"Journal File System Error--Call NIC Center");          0730
   RETURN(flagut(0, $tstfg));                                       0731
   END.
                                                                    0732
(lockjo)PROC(type);                                                 0733
   %Type = 1 for bad file lock (flag 1), and 0 for jlock (flag 0);
                                                                    0734
   %                                                                0735
   IF type > 1 THEN err($"Illegal Flag Number");                    0736
   flagut(type, $setfg);                                            0737
   specttyout(0, $"Journal Locked");                                0738
   specttyout(30B, $"Journal Locked");                              0739
   RETURN;                                                          0740
   END.                                                             0741
(unlkjo)PROC(type);                                                 0742
   %Type indates flag to be unlocked... -1 means all%               0743
   IF type NOT IN [-1, 1] THEN err($"Illegal Flag Number");         0744
   IF type # 1 THEN flagut(0, $rstfg);                              0745
   IF type # 0 THEN flagut(1,$rstfg);                               0746
   RETURN;                                                          0747
   END.                                                             0748
%....Journal submission....%                                        0749
   (jsubmit) %do the Journal submission or secondary distribution%
   PROCEDURE;                                                       0750
      %This procedure submits an itemm to the Journal by creating a
      properly formatted file in the directory <tejournal>.%        0751
      LOCAL  rnumstid, numstid, fl, traping, capsav, connflag, stid,
      orgstr;                                                       0752
      LOCAL TEXT POINTER z1, z2, z3, z4, z5, z6, where;             0767
      LOCAL STRING                                                  0754
         number[20],      %journal number%                         0755
         rfcnum[20],      %RFC number%                             0756
         itemtype[20],    %item type%                              0757
         linkaddr[150],   %link to location for link insert%       0758
         fnamestr[150],   %file name string%                       0759
         goodids[500],    %list of good idents%                    0760
         badids[500],     %list of bad idents%                     0761
         workstr[500],    %work string%                            0762
         accesslist [1000], %access list%                          01197
         dpassw[15],      %password string%                        0763
         dirnam[50],      %directory name%                         0764
         dfilstr[150],    %tejournal file name%                    0765
         mfname[150];    %message file name%                       0766
      REF fl, orgstr;                                               0753
      jolock(); %Check to make sure it will work -- does not return
```

```
unless things are cool%                                                 0768
%set stid's to null%                                                    0769
   rnumstid ← numstid ← orgstid;                                        0770
capsav ← 0;                                                             0771
traping ← connflag ← FALSE;                                             0772
&orgstr ← 0;                                                           01289
ON SIGNAL                                                               0773
   ELSE                                                                 0774
      BEGIN %close any open files%                                      0775
      ON SIGNAL ELSE;                                                   0776
      conjdir(FALSE);                                                   0777
      IF numstid.stfile THEN close(numstid.stfile := 0);                0778
      IF rnumstid.stfile THEN close(rnumstid.stfile := 0);              0779
      IF traping := FALSE THEN notrapcc(capsav);                        0781
      IF connflag := FALSE THEN %connect back to user's
      directory%                                                        0782
         condir($dirnam,$dpassw,$dirnam,$dpassw);                       0783
      % Check orgstr; if empty, simply release string.  If not,
      reset the origin statement to its original state. %              01290
         IF &orgstr THEN                                               01291
            BEGIN                                                      01292
            IF orgstr.L THEN                                           01293
               BEGIN                                                   01294
               % Restore original origin statement. %                  01295
                  ST jwpl ← *orgstr*;                                  01296
               END;                                                    01297
            freestring(&orgstr, $dspblk);                              01298
            END;                                                       01299
      END;                                                              0784
IF NOT idfno THEN idfno ← loaidfil();                                   0785
% Get string to be used for origin statement restoration. %            01300
   &orgstr ← getstring(2000, $dspblk);                                 01301
   *orgstr* ← SF(jwpl) SE(jwpl);                                       01302
%check for non-null source%                                            01249
   stid ← getsub(jwpl);                                                01250
   IF getnxt(stid#]   endfil AND ( FIND SF(stid) *nullsource*
   ENDCHR ) THEN                                                       01251
      werr($"Nothing to send!");                                       01252
%check ident lists%                                                     0786
   getjauthor($workstr);                                                0787
      IF *workstr* # *initsr* THEN %must check it%                      0788
         BEGIN                                                          0789
         *goodids* ← *workstr*;                                         0790
         ckidlist($goodids, $badids, idfno);                            0791
         IF badids.L THEN %tell user the bad news%                      0792
            BEGIN                                                       0793
            IF goodids.L THEN setjauthor($goodids)                      0794
            ELSE setjauthor($initsr);                                   0795
            *badids* ← "Illegal ident(s) [mail not sent]: ",
            *badids*;                                                   0796
            werr($badids);                                              0797
            END;                                                        0798
         IF *goodids* # *workstr* THEN setjauthor($goodids);            0799
         END;                                                           0800
   getjaction($workstr);                                                0801
      IF workstr.L THEN %must check it%                                 0802
```

```
         BEGIN                                                  0803
         *goodids* ← *workstr*;                                 0804
         ckidlist($goodids, $badids, idfno);                    0805
         IF badids.L THEN %tell user the bad news%              0806
            BEGIN                                               0807
            setjaction($goodids);                               0808
            *badids* ← "Illegal ident(s) /mail not sent/: ",
            *badids*;                                           0809
            werr($badids);                                      0810
            END;                                                0811
         IF *goodids* # *workstr* THEN setjaction($goodids);    0812
         END;                                                   0813
   getjinfo($workstr);                                          0814
      IF workstr.L THEN %must check it%                         0815
         BEGIN                                                  0816
         *goodids* ← *workstr*;                                 0817
         ckidlist($goodids, $badids, idfno);                    0818
         IF badids.L THEN %tell user the bad news%              0819
            BEGIN                                               0820
            setjinfo($goodids);                                 0821
            *badids* ← "Illegal ident(s) /mail not sent/: ",
            *badids*;                                           0822
            werr($badids);                                      0823
            END;                                                0824
         IF *goodids* # *workstr* THEN setjinfo($goodids);      0825
         END;                                                   0826
%set up strings for later use%                                  0827
   getjrfc($rfcnum);                                            0828
   getjnumber($number);                                         0829
      IF NOT rfcnum.L AND NOT number.L THEN  %kludge for now
      until fix background guy%                                 0830
         BEGIN                                                  0831
         *number* ← "1234";                                     0832
         setjnumber($"1234");                                   0833
         END                                                    0834
      ELSE IF *number* # "1234" THEN %check it%                 0835
         BEGIN                                                  0836
         getjauthor($workstr);                                  0837
         IF NOT FIND SF(*workstr*) [*initsr*] THEN              0838
            *workstr* ← *initsr*, SP, *workstr*;                0839
         conjdir(TRUE);                                         0840
         numstid ← openlock(0, jflname($"tcnumbers"));          0841
         IF rfcnum.L THEN %check rfc number and journal number%
                                                                0842
            BEGIN                                               0843
            rnumstid ← openlock(0, jflname($"Rfcnumbers"));     0844
            ckrfcnum(1, $rfcnum, $number, $workstr, rnumstid,
            numstid);                                           0845
            END                                                 0846
         ELSE ckcnum(1, $number, $workstr, numstid);            0847
         conjdir(FALSE);                                        0848
         END;                                                   0849
   getjlink($linkaddr);                                         0850
   getjtype($itemtype);                                         0851
%fill in time and date (of submission) fields%                  0852
   *datesr* ← NULL;                                             0853
```

```
    !gtad;                                                              0854
    dtfrmt(r1, $datesr);                                                0855
    IF NOT FIND jwp1 > [";;;;"] [ "(J" / "(j" ] [';] ↑p1 ←p1
    ["Author(s):"] ['//] ↑p3 [';] ↑p4 ←p4 THEN                          0856
       werr($"Journal Header improperly formatted (mail not
       sent)");                                                         0857
    IF rfcnum.L THEN                                                    0858
       ST jwp1 ← SF(p1) p1, SP, *datesr*, "; .HJOURNAL=", '",
       "NWG/RFC# ", *rfcnum*, ".SPLIT;",  p3 p4, SP, *datesr*, "
       ", *number*, '",  p1 SE(p1)                                      0859
    ELSE                                                                0860
       ST jwp1 ← SF(p1) p1, SP, *datesr*, "; .HJOURNAL=", '", p3
       p4, SP, *datesr*, "  ", *number*, '",  p1 SE(p1);               0861
%build access list if private document%                                 0862
    IF rdprvsts (jwp1.stfile) = $psprivate THEN                         0705
       BEGIN                                                           01195
       accesslist.L ← 0;                                                0707
       clnidist (getjaction ($workstr), $accesslist);                   0708
       clnidist (getjinfo ($workstr), $accesslist);                     0709
       clnidist (getjauthor ($workstr), $accesslist);                   0710
       clnidist (getjclerk ($workstr), $accesslist);                    0711
       setjaxcess ($accesslist);                                        0713
       END;                                                            01196
%convert subcollections and distribution list -- should be done
when tejournal file is loaded by background%                            0864
    convjdist();                                                        0865
    convjsubcol(idfno);                                                 0866
%decide between storage as message or file%                             0867
    IF FIND jwp1 > [";;;; "] ↑p1 CH ↑p2 THEN                            0868
       IF rdprvsts (jwp1.stfile) = $psprivate                          01199
       OR getnxt (getnxt (jworkstid)) # endfil                         01200
       % OR NOT jtypchk (2000, jworkstid) % THEN                       01198
          ST p1 p2 ← 'F                                                 0869
       ELSE ST p1 p2 ← 'M;                                             01202
%conn. to TEJOURNAL directory and update the workfile into it%
                                                                        0870
    IF NOT number.L OR *number* = "1234" THEN  %build
    deferred-number file name%                                          0871
       BEGIN                                                            0872
       FIND SF(*datesr*) [1$D] ↑z2 < D ↑z1 > [':] ↑z5 <CH ↑z4 $D
       ↑z3 z5 > $D ↑z6;                                                 0873
       *dfilstr* ← 'D, z1 z2, z3 z4, z5 z6, *initsr*, ".NLP";           0874
       END                                                              0875
    ELSE  %build pre-assigned number file name%                         0876
       *dfilstr* ← 'J, *number*, ".NLN";                                0877
    *dirnam* ← NULL;                                                    0878
    &f1 ← fintadr(jworkstid.stfile);                                    0879
    jfnstr(r1.florig, $fnamestr);                                       0880
    traping ← TRUE;                                                     0881
    capsav ← trapcc();                                                  0882
    connflag ← TRUE;                                                    0883
    condir(IF jdebug THEN $"IRBY" ELSE $"TEJOURNAL",IF jdebug THEN
    $"BLI" ELSE $jnlpsw,$dirnam,$dpassw);                               0884
    updtfl(jworkstid.stfile, newversion, $dfilstr);                     0885
    condir($dirnam,$dpassw,$dirnam,$dpassw);                            0886
    connflag ← FALSE;                                                   0887
```

```
    IF traping:=FALSE THEN notrapcc(capsav);                          0888
    fl.flnoclos ← FALSE;                                              0890
    close(jworkstid.stfile);                                          0891
    jworkstid ← 0;                                                    0892
    % Dispose of orgstr space. %                                     01303
       freestring(&orgstr:=0, $dspblk);                               01304
    % Zap the Journal workfile. %                                    01305
       ON SIGNAL ELSE                                                 0893
          BEGIN %must create it%                                      0894
          ON SIGNAL ELSE;                                             0895
          jworkstid ← openwk(0, $fnamestr);                           0896
          GOTO jsubok;                                                0897
          END;                                                        0898
       jworkstid ← orgstid;                                           0899
       jworkstid.stfile ← open(0, $fnamestr);                         0900
       resetf(jworkstid.stfile);                                      0901
       (jsubok):  %initialize the file%                               0902
       &fl ← flntadr(jworkstid.stfile);                               0903
       fl.flnoclos ← TRUE;                                            0904
       initjwork();  %zap the Journal workfile%                       0905
       FIND SF(jworkstid) ↑jwpl;                                      0906
ON SIGNAL                                                             0909
   ELSE                                                               0910
       BEGIN %close any open files%                                   0911
       ON SIGNAL ELSE;                                                0912
       conjdir(FALSE);                                               01306
       IF numstid.stfile THEN close(numstid.stfile := 0);             0913
       IF rnumstid.stfile THEN close(rnumstid.stfile := 0);           0914
       IF idfno THEN close(idfno := 0);                               0915
       IF traping := FALSE THEN notrapcc(capsav);                     0916
       END;                                                           0917
IF number.L AND *number* # "1234" THEN                                0907
   BEGIN                                                              0908
   conjdir(TRUE);                                                     0918
   IF rfcnum.L THEN                                                   0919
      BEGIN %release RFC number%                                      0920
      usedcnum($rfcnum, rnumstid);                                    0921
      END;                                                            0922
   usedcnum($number, numstid); %release used journal number%          0923
   conjdir(FALSE);                                                    0924
   END;                                                               0925
IF linkaddr.L THEN                                                    0926
   BEGIN                                                              0927
   CASE *itemtype*[1] OF                                              0928
      = 'H: *mfname* ← "(Hard Copy -- Journal, ", *number*, ",)";
                                                                      0929
      ENDCASE *mfname* ← "( Journal, ", *number*, ", 1:w)" ;          0930
   %now insert the link%                                              0931
      FIND SF(*linkaddr*) ↑z1 SE(*linkaddr*) ↑z2;                     0932
      caddexp($z1, $z2, lda(), $where);                               0933
         %where should the starting point of the expression
         be????  we leave it undefined%                              01308
      IF where = endfil THEN                                          0934
         BEGIN                                                        0935
         *linkaddr* ← "Unable to insert link ", *mfname*, " at ",
         *linkaddr*;                                                  0936
```

```
            dismes(1, $linkaddr);                                          0937
            END                                                            0938
         ELSE                                                              0939
            BEGIN                                                          0940
            FIND SF(*mfname*) ↑z1 SE(*mfname*) ↑z2;                        0941
            dismes(1, $linkaddr);                                          0942
            cinstex($where, $z1, $z2, TRUE);                               0943
            END;                                                           0944
      END;                                                                 0945
   dismes(1, $"Completed");                                                0946
   IF idfno THEN close(idfno := 0);                                        0947
   IF numstid.stfile THEN close(numstid.stfile := 0);                      0948
   IF rnumstid.stfile THEN close(rnumstid.stfile := 0);                    0949
   RETURN;                                                                 0950
   END.
                                                                           0951
(secdist)PROCEDURE(number, distlist, actionflag);  %execute forward
journal item request%                                                      01309
   %Accepts a string containing a catalog number, and a string
   containing an identlist and a flag (true if action, false if
   info-only).                                                             01310
      Distributes the indicated Journal document to the persons on
      the list%                                                            01311
   LOCAL char, jcstid, fl, traping, capsav, connflag;                      01312
   REF number, distlist, fl;                                               01315
   LOCAL TEXT POINTER z1, z2, z3, z4;                                      01313
   LOCAL STRING                                                            01452
      badids[500],     %list of bad idents%                               01459
      dpassw[15],      %password string%                                  01462
      dirnam[50],      %directory name%                                   01463
      dfilstr[150],    %tejournal file name%                              01464
      idstr[25],       %to hold an ident%                                 01469
      comm[300],       %to hold the comment associated with an ident%
                                                                          01468
      fnamestr[150],   %file name string%                                 01496
      tempsr[400];     %temporary work string%                            01467
   jcstid  ← orgstid;                                                     01316
   capsav ← 0;                                                            01474
   traping ← connflag ← FALSE;                                            01473
   ON SIGNAL                                                              01475
      ELSE                                                                01476
         BEGIN %close any open files%                                     01477
         ON SIGNAL ELSE;                                                  01478
         conjdir(FALSE);                                                  01479
         IF traping := FALSE THEN notrapcc(capsav);                       01482
         IF connflag := FALSE THEN %connect back to user's
         directory%                                                       01483
            condir($dirnam,$dpassw,$dirnam,$dpassw);                      01484
         close(jcstid.stfile := 0);                                       01317
         END;                                                             01495
   IF NOT (jcstid ← gtcatent (&number, 0, 0, FALSE)) THEN                 01318
      werr ($"No such document");                                         01319
   IF NOT vrprvacc (0, 0, jcstid, $initsr) THEN                           01320
      werr ($"Private document; access denied to you");                   01321
   IF NOT idfno THEN idfno ← loaidfil();                                  01342
   %check distribution list%                                              01357
```

```
    ckidlist($distlist, $badids, idfno);                                    01361
    IF badids.L THEN %tell user the bad news%                               01362
       BEGIN                                                                01363
       *badids* ← "Illegal ident(s) [mail not forwarded]: ",
       *badids*;                                                            01365
       werr($badids);                                                       01366
       END;                                                                 01367
 %put in ACTION or INFO-only designation%                                   01409
    IF actionflag THEN %process as action list%                             01373
       *dirnam* ← " [ ACTION ] "                                            01471
    ELSE %process as information-only list%                                 01388
       *dirnam* ← " [ INFO-ONLY ] ";                                        01472
    *tempsr* ← *distlist*;                                                  01375
    *distlist* ← NULL;                                                      01470
    %add ( [ ACTION ] )  or ( [INFO-ONLY] ) to each ident; must
    agree with format checked in (nls, jnldel, distdl)%                     01377
       FIND SF(*tempsr*) ↑z1;                                               01378
       WHILE (FIND z1 $NP $(',) $NP ↑z2 1$(LD / '↑ / '& / '-) ↑z1)
       DO                                                                   01379
          BEGIN                                                             01380
          *idstr* ← +z2 z1;                                                 01381
          IF FIND z1 $NP '( ↑z2 ([')] ↑z1 ↑z3 ←z3 /[ENDCHR] ↑z1
          ↑z3) THEN                                                         01382
             *comm* ← z2 z3                                                 01383
          ELSE *comm* ← NULL;                                               01384
          *distlist* ← *distlist*, *idstr*, '(, *dirnam*, *comm*,
          '), SP;                                                           01385
          END;                                                              01386
    %terminate with semicolon%                                              01497
       *distlist* ← *distlist*, '; ;                                        01498
 ST jworkstid ← ";;;; S ", SF(jcstid) SE(jcstid);                           01326
 z2 ← getsub(jcstid);                                                       01327
 z2 ← ccopsta(jworkstid, levdown, z2, FALSE, 0);                            01328
 z2 ← cis(z2, &distlist, $succdir);                                         01330
 cis(z2, $initsr, $succdir);                                                01331
 %update to TEJOURNAL%                                                      01332
    *dfilstr* ← 'S, *number*, ".NLN";                                       01333
    *dirnam* ← NULL;                                                        01334
    &fl ← fintadr(jworkstid.stfile);                                        01421
    jfnstr(fl.florig, $fnamestr);                                           01422
    traping ← TRUE;                                                         01423
    capsav ← trapcc();                                                      01424
    connflag ← TRUE;                                                        01425
    condir(IF jdebug THEN $"IRBY" ELSE $"TEJOURNAL",IF jdebug THEN
    $"ELI" ELSE $jnlpsw,$dirnam,$dpassw);                                   01426
    updtfl(jworkstid.stfile, newversion, $dfilstr);                         01427
    condir($dirnam,$dpassw,$dirnam,$dpassw);                                01428
    connflag ← FALSE;                                                       01429
    IF traping:=FALSE THEN notrapcc(capsav);                                01430
    fl.flnoclos ← FALSE;                                                    01431
    close(jworkstid.stfile);                                                01432
    jworkstid ← 0;                                                          01433
    % Zap the Journal workfile. %                                          01436
       ON SIGNAL ELSE                                                       01437
          BEGIN %must create it%                                            01438
          ON SIGNAL ELSE;                                                   01439
```

```
            jworkstid ← openwk(0, $fnamestr);                          01440
            GOTO jsubok;                                               01441
            END;                                                       01442
         jworkstid ← orgstid;                                          01443
         jworkstid.stfile ← open(0, $fnamestr);                        01444
         resetf(jworkstid.stfile);                                     01445
         (seedok):  %initialize the file%                              01446
         &fl ← flntadr(jworkstid.stfile);                              01447
         fl.flnoclos ← TRUE;                                           01448
         initjwork();  %zap the Journal workfile%                      01449
         FIND SF(jworkstid) ↑jwpl;                                     01450
   dismes(1, $"Completed");                                            01339
   close(jcstid.stfile := 0);                                          01340
   RETURN END.
                                                                       01341
%....Network Journal Procedures....%                                    0952
  (njs) %network journal submission%
  PROCEDURE (acs);                                                      0953
   %----------%                                                         0954
   LOCAL ct, nxtstid, substid, injfn, titlestid, boundstid, tally,
   tailstrt, tailend;                                                   0955
   LOCAL STRING                                                         0956
      infile [40], control [200], authlist [200], distlist [200],
      clerk [20], title [200], errstr [500], level [30], workfile
      [40];                                                             0957
   LOCAL TEXT POINTER tp1, tp2, tp3, tp4, tp5;                          0958
   REF acs;                                                             0959
   %----------%                                                         0960
   %avoid scheduler pit%                                                0961
      r1 ← 400000B;                                                     0962
      r2 ← 303B;                                                        0963
      ! JSYS spriw;                                                     0964
   %open, read, and close control file%                                 0965
      ON SIGNAL ELSE                                                    0966
         hiortn (cat, $"Control file error: ", sysmsg, 0);              0967
      r1 ← acs [7]: ! HRRZS 1; injfn ← r1;                              0968
      IF NOT sysopen (injfn, read, chrtyp, $errstr) THEN                0969
         hiortn (cat, $"Control file can't be opened: ", $errstr,
         0);                                                            0970
      IF NOT isread (injfn, $control, CR) THEN                          0971
         hiortn (cat, $"Control file can't be read", 0, 0);             0972
      sysclose (injfn, $errstr);                                        0973
   %determine conversion type%                                          0974
      ct ← 'S;                                                          0975
      IF FIND SF(*control*) [';] ↑tp1 ←tp1 ↑tp2 CH ↑tp3 tp2 THEN        0976
         BEGIN                                                          0977
         CASE (ct ← READC .A (177B-SP)) OF                              0978
            ='S, %insert sequential%                                    0979
            ='H, %heuristic w/o prev right justfiy%                     0980
            ='J, %heuristic w/ prev right justify%                      0981
            ='A, %insert assembler w/ structure%                        0982
            ='M: %insert assembler w/o structure%                       0983
               NULL;                                                    0984
            ENDCASE                                                     0985
               hiortn (cat, $"Valid conversion types are A, M, S, H,
               and J", 0, 0);                                           0986
```

```
      ST tp1 tp3 ← NULL;                                              0987
      END;                                                            0988
   ct ← 'S; %force sequential conversion for now%                     0989
%verify and save clerk, author(s), addressee(s)%                      0990
   ON SIGNAL ELSE                                                     0991
      hiortn (cat, $"Ident verification error: ", sysmsg, 0);         0992
   IF NOT FIND SF(*control*) $(SP/',) ↑tp1 1$(LD/'-) ↑tp2 ['/]
   ↑tp3 ←tp3 ↑tp4 [CR] ↑tp5 ←tp5 THEN                                0993
      hiortn (cat, $"Ident list format error", 0, 0);                 0994
   *clerk* ← + tp1 tp2;                                               0995
   *initsr* ← *clerk*;                                                0996
   *authlist* ← + tp1 tp3;                                            0997
   *distlist* ← + tp4 tp5;                                            0998
   IF NOT njsverids ($clerk, $authlist, $distlist, $errstr)           0999
      THEN hiortn (cat, $"No such ident(s): ", $errstr, 0);           01000
   hioco (ok, $acs);                                                  01001
%fetch name of sequential text file%                                  01002
   ON SIGNAL ELSE                                                     01003
      hiortn (cat, $"Work file init. error: ", sysmsg, 0);            01004
   r1 ← acs [7]; ! HRRZS 1; injfn ← r1;                               01005
   jfnstr (injfn, $infile);                                           01006
%initialize work file%                                                01007
   *workfile* ← "<SYSTEM>[SEND-MAIL].", *initsr*, ";1;P707000";
                                                                      01008
   ON SIGNAL ELSE                                                     01009
      BEGIN %must create it%                                          01010
      ON SIGNAL ELSE                                                  01011
         hiortn (cat, $"Work file init. error: ", sysmsg, 0);
                                                                      01012
      jworkstid ← openwk (0, $workfile);                              01013
      GOTO okay;                                                      01014
      END;                                                            01015
   jworkstid ← orgstid;                                               01016
   jworkstid.stfile ← open (0, $workfile);                            01017
   initjwork ();                                                      01018
   FIND SF(jworkstid) ↑jwp1;                                          01019
   (okay): ON SIGNAL ELSE                                             01020
      hiortn (cat, $"Work file init. error: ", sysmsg, 0);            01021
   setjauthor ($authlist);                                            01022
   setjaction ($distlist);                                            01023
%insert text of message/file%                                         01024
   IF (substid ← getsub (jworkstid)) # jworkstid THEN                 01025
      cdelgro (substid, getail (substid), FALSE, 0);                  01026
   level.L ← 0;                                                       01027
   CASE ct OF                                                         01028
      ='S: %insert sequential%                                        01029
         inseq (jworkstid, $level, $infile, tenfil);                  01030
      ='H: %heuristic insert sequential w/o prev right justify%
                                                                      01031
         inseqn (jworkstid, $level, $infile, FALSE);                  01032
      ='J: %heuristic insert sequential w/ prev right justify%
                                                                      01033
         inseqn (jworkstid, $level, $infile, TRUE);                   01034
      ='A: %insert assembler w/ Structure%                            01035
         inseq (jworkstid, $level, $infile, assfil);                  01036
      ='M: %insert assembler w/o Structure%                           01037
```

```
            inseq (jworkstid, $level, $infile, macfil);                    01038
         ENDCASE;                                                          01039
   %extract the title (if any)%                                            01040
      titlestid ← boundstid ← tailstrt ← tally ← 0;                        01041
      IF ((nxtstid ← getnxt (jworkstid)) # endfil) THEN                    01042
         WHILE ((nxtstid # jworkstid) AND ((tally ← tally+1) <= 15))
         DO                                                                01043
            BEGIN                                                          01044
            IF FIND SF(nxtstid) ↑tp1 [':] ↑tp2 ←tp2 $SP ↑tp3
            ([CR/EOL] < CH / SE(nxtstid)) $SP ↑tp4 THEN                    01045
               IF (tp2 [1] - tp1 [1]) <= title.M THEN                      01046
                  BEGIN                                                    01047
                  *title* ← + tp1 tp2;                                     01048
                  IF *title* = "RE"                                        01049
                  OR *title* = "TITLE"                                     01050
                  OR *title* = "SUBJECT" THEN                              01051
                     BEGIN                                                 01052
                     titlestid ← nxtstid;                                  01053
                     *title* ← tp3 tp4;                                    01054
                     IF title.L THEN setjtitle ($title);                   01055
                     END;                                                  01056
                  END;                                                     01057
            IF FIND SE(nxtstid) "- - - -" THEN                             01058
               BEGIN                                                       01059
               boundstid ← nxtstid;                                        01060
               WHILE (NOT getftl (nxtstid)) DO                             01061
                  BEGIN                                                    01062
                  nxtstid ← getsuc (nxtstid);                              01063
                  IF FIND SE(nxtstid) "-------" THEN                       01064
                     BEGIN                                                 01065
                     tailstrt ← nxtstid;                                   01066
                     WHILE (NOT getftl (nxtstid)) DO                       01067
                        nxtstid ← getsuc (nxtstid);                        01068
                     tailend ← nxtstid;                                    01069
                     EXIT LOOP 2;                                          01070
                     END;                                                  01071
                  END;                                                     01072
               EXIT LOOP;                                                  01073
               END;                                                        01074
            IF NOT FIND SF(nxtstid) [PT] THEN                              01075
               BEGIN                                                       01076
               boundstid ← nxtstid;                                        01077
               EXIT LOOP;                                                  01078
               END;                                                        01079
            nxtstid ← getsuc (nxtstid);                                    01080
            END;                                                           01081
      ON SIGNAL ELSE GOTO done;                                            01082
      IF tailstrt THEN                                                     01083
         cdelgrc (tailstrt, tailend, FALSE, 0);                            01084
      IF boundstid THEN                                                    01085
         cdelgrc (getnxt (jworkstid), boundstid, FALSE, 0);                01086
      IF titlestid THEN cdelsta (titlestid, FALSE, 0);                     01087
      (done): ON SIGNAL ELSE                                               01088
         hiortn (cat, $"JWORK file init. error: ", sysmsg, 0);             01089
   %null document?%                                                        01090
      IF getnxt (jworkstid) = endfil THEN                                  01091
```

```
        cis (jworkstid, $nullscurce, $"d");                          01092
    %journalize%                                                     01097
        ON SIGNAL ELSE                                               01098
            hiortn (cat, $"Submission error: ", sysmsg, 0);          01099
        jsubmit ();                                                  01100
        hiortn (ok, $"Journalized", 0, 0);                           01101
        RETURN;                                                      01102
        END.                                                         01103
(njsverids) %verify ident lists%
PROCEDURE (clerk, authlist, distlist, errlst);                       01104
    %----------%                                                    01105
    LOCAL idfileno;                                                  01106
    LOCAL STRING ident [30];                                         01107
    LOCAL TEXT POINTER tp1, tp2;                                     01108
    REF clerk, authlist, distlist, errlst;                           01109
    %----------%                                                    01110
    idfileno ← open (0, jflname ($"Identfile"));                     01111
    ON SIGNAL ELSE IF idfileno THEN                                  01112
        sigclose (idfileno := 0);                                    01113
    errlst.L ← 0;                                                    01114
    %verify clerk%                                                   01115
        IF NOT oldid (&clerk, idfileno) THEN                         01116
            *errlst* ← *errlst*, *clerk*, SP;                        01117
    %verify authors%                                                 01118
        FIND SF(*authlist*) ↑tp2;                                    01119
        WHILE (FIND tp2 > $(SP/',) ↑tp1 1$(LD/'-) ↑tp2) DO           01120
            BEGIN                                                    01121
            *ident* ← tp1 tp2;                                       01122
            IF NOT oldid ($ident, idfileno) THEN                     01123
                *errlst* ← *errlst*, *ident*, SP;                    01124
            END;                                                     01125
    %verify addressees%                                              01126
        FIND SF(*distlist*) ↑tp2;                                    01127
        WHILE (FIND tp2 > $(SP/',) ↑tp1 1$(LD/'-) ↑tp2) DO           01128
            BEGIN                                                    01129
            *ident* ← tp1 tp2;                                       01130
            IF NOT oldid ($ident, idfileno) THEN                     01131
                *errlst* ← *errlst*, *ident*, SP;                    01132
            END;                                                     01133
    IF errlst.L THEN BUMP DOWN errlst.L;                             01134
    close (idfileno := 0);                                           01135
    RETURN (NOT errlst.L);                                           01136
    END.                                                             01137
(hiortn) %return to calling fork%
PROCEDURE (outcome, hdr, body, trlr);                                01138
    %----------%                                                    01139
    LOCAL STRING msg [500];                                          01140
    REF hdr, body, trlr;                                             01141
    %----------%                                                    01142
    msg.L ← 0;                                                       01143
    IF &hdr THEN *msg* ← *msg*, *hdr*;                               01144
    IF &body THEN *msg* ← *msg*, *body*;                             01145
    IF &trlr THEN *msg* ← *msg*, *trlr*;                             01146
    dismes (2, $msg);                                                01147
    RETURN;                                                          01148
```

```
      *msg* ← *msg*, 0;                                                    01149
      r2 ← chbptr (0) + $msg;                                              01150
      r1 ← outcome;                                                        01151
      ! haltf;                                                             01152
      END.                                                                 01153
   (hiocc) %co-routine return to calling fork%
   PROCEDURE (outcome, acs);                                               01154
      %----------%                                                         01155
      LOCAL STRING filename [40], error [200?];                            01156
      REF acs;                                                             01157
      %----------%                                                         01158
      *filename* ← "NJS.SRC";                                              01159
      acs [7] ← sgtjfn (1000000000000B, $filename, $error);                01160
      RETURN;                                                              01161
      r1 ← outcome;                                                        01162
      r2 ← 0;                                                              01163
      r4 ← $acs [7]; ! MOVE 3,7;                                           01164
      ! haltf;                                                             01165
      ! EXCH 3,7; ! MOVEM 3,(4);                                           01166
      RETURN;                                                              01167
      END.                                                                 01168
% miscellaneous  %                                                        01169
   (jtypchk) PROC (maxchars, jstid); % check number of chars in a
   jworkfile  %                                                            01170
      % jstid is statement preceeding the body of the submssion %          01171
      % Returns TRUE if <= max %                                          01172
      LOCAL  stid, size;                                                   01173
      size ← 0;                                                            01174
      stid ← jstid;                                                        01175
      WHILE (stid ← getnxt(stid)) # endfil AND size <= maxchars DO         01176
         size ← size + getstsize(stid);                                    01177
      RETURN(IF size > maxchars THEN FALSE ELSE TRUE);                     01178
      END.

                                                                           01179
   (jlog)PROC(number, typest, identsr);                                    01180
      %Type message to Journal Logging tty.                                01181
         Number = Number                                                   01182
         typest is a string which describes use of number                  01183
         identsr is the ident of author. etc.                              01184
      %                                                                    01185
      REF number, typest, identsr;                                         01186
      LOCAL STRING tempsr[200];                                            01187
      *tempsr* ← *number*, SP, *typest*, SP, *identsr*, SP;                01188
      !JSYS gtad;                                                          01189
      dtfrmt(r1, $tempsr); %Add the date/time%                             01190
      specttyout(10B, $tempsr);                                            01191
      RETURN;                                                              01192
      END.                                                                 01193
FINISH                                                                     01194
```

```
< NLS. CSYNGEN.NLS;4, >;  4-OCT-74 15:20 KEV ;;;;
FILE csyngen % L10 <rel-nls>csyngen %% (L10,) (rel-nls,csyngen.rel,) %
                                                                      0813
(ckcoprul) PROCEDURE  % copy rule %                                    028
   % FORMALS %                                                         029
      (destination, lvl, azrule);                                      030
   LOCAL i, savstid;                                                   031
   omode ← 1;                                                          032
   cinstid ← destination;                                              033
   ilevel ← lvl;                                                       034
   *lit* ← NULL;                                                       035
   pstkp ← 0;                                                          036
   nprint( $lit, azrule, TRUE, FALSE);                                 037
   savstid ← cinstid;                                                  038
   ilevel ← levdown;                                                   039
   FOR i ← 0 UP UNTIL >= pstkp DO                                      040
      BEGIN                                                            041
      IF pstk[i+1].LH THEN                                             042
         *lit* ← *[pstk[i+1].LH]*, "="                                 043
      ELSE                                                             044
         BEGIN                                                         045
         gtruln( pstk[i+1].RH, $lit );                                 046
         *lit* ← *lit*, " =";                                          047
         END;                                                          048
      onode( $lit );                                                   049
      ilevel ← levdown;                                                050
      *lit* ← NULL;                                                    051
      nprint( $lit, pstk[i+1].RH, TRUE, FALSE);                        052
      ilevel ← levup;                                                  053
      END;                                                             054
   RETURN( savstid );                                                  055
   END.
                                                                       056
   %%                                                                  057
```

```
(ckcopcmd) PROCEDURE  % copy command %                                  058
   % FORMALS %                                                          059
      (destination, lvl, azcmnd);                                       060
   LOCAL i, savstid;                                                    061
   omode ← 1;                                                           062
   cinstid ← destination;                                               063
   ilevel ← lvl;                                                        064
   *lit* ← NULL;                                                        065
   pstkp ← 0;                                                           066
   nprint( $lit, azcmnd, FALSE, FALSE);                                 067
   savstid ← cinstid;                                                   068
   ilevel ← levdown;                                                    069
   FOR i ← 0 UP UNTIL >= pstkp DO                                       070
      BEGIN                                                             071
      IF pstk[i+1].LH THEN                                              072
         *lit* ← *[pstk[i+1].LH]*, "="                                  073
      ELSE                                                              074
         BEGIN                                                          075
         gtruln( pstk[i+1].RH, $lit );                                  076
         *lit* ← *lit*, " =";                                           077
         END;                                                           078
      onode( $lit );                                                    079
      ilevel ← levdown;                                                 080
      *lit* ← NULL;                                                     081
      nprint( $lit, pstk[i+1].RH, TRUE, FALSE);                         082
      ilevel ← levup;                                                   083
      END;                                                              084
   RETURN( savstid );                                                   085
   END.
                                                                        086
   %%                                                                  087
```

```
(ckcopsub) PROCEDURE  % copy subsystem %                                    088
   % FORMALS %                                                              089
      (destination, lvl, azsubn);                                           090
   LOCAL anode;                                                             091
   REF anode;                                                               092
   &anode ← (azsubn).dptrun;                                                093
   cinstid ← destination;                                                   094
   ilevel ← lvl;                                                            095
   WHILE &anode DO                                                          096
      BEGIN                                                                 097
      cinstid ← ckcopcmd( cinstid, ilevel, &anode);                         098
      ilevel ← levsuc;                                                      099
      &anode ← anode.alternative;                                          0100
      END;                                                                 0101
   RETURN( cinstid );                                                      0102
   END.
                                                                           0103
   %%                                                                      0104
```

```
(ckshwrul) PROCEDURE  % show rule %                                    0105
   % FORMALS %                                                         0106
      (azrule);                                                        0107
   LOCAL i;                                                            0108
   omode ← 3;                                                          0109
   *lit* ← NULL;                                                       0110
   pstkp ← 0;                                                          0111
   nprint( $lit, azrule, TRUE, FALSE);                                 0112
   FOR i ← 0 UP UNTIL >= pstkp DO                                      0113
      BEGIN                                                            0114
      IF pstk[i+1].LH THEN                                             0115
         *lit* ← "    ", *[pstk[i+1].LH]*, "="                         0116
      ELSE                                                             0117
         BEGIN                                                         0118
         gtruln( pstk[i+1].RH, $lit );                                 0119
         *lit* ← "    ", *lit*, " =";                                  0120
         END;                                                          0121
      onode( $lit );                                                   0122
      *lit* ← "      ";                                                0123
      nprint( $lit, pstk[i+1].RH, TRUE, FALSE);                        0124
      END;                                                             0125
   RETURN;                                                             0126
   END.
                                                                       0127
   %%                                                                  0128
```

```
(ckshwcmd) PROCEDURE  % show command %                                  0129
   % FORMALS %                                                          0130
      (azcmnd);                                                         0131
   LOCAL i;                                                             0132
   omode ← 3;                                                           0133
   *lit* ← NULL;                                                        0134
   pstkp ← 0;                                                           0135
   nprint( $lit, azcmnd, FALSE, FALSE);                                 0136
   FOR i ← 0 UP UNTIL >= pstkp DO                                       0137
      BEGIN                                                             0138
      IF pstk[i+1].LH THEN                                              0139
         *lit* ← "    ", *(pstk[i+1].LH)*, "="                         0140
      ELSE                                                              0141
         BEGIN                                                          0142
         struln( pstk[i+1].RH, $lit );                                  0143
         *lit* ← "    ", *lit*, " =";                                   0144
         END;                                                           0145
      onode( $lit );                                                    0146
      *lit* ← "       ";                                                0147
      nprint( $lit, pstk[i+1].RH, TRUE, FALSE);                         0148
      END;                                                              0149
   RETURN;                                                              0150
   END.
                                                                        0151
   %%                                                                  0152
```

```
(ckshwsub) PROCEDURE  % show subsystem %                                0153
   % FORMALS %                                                          0154
      (azsubn);                                                         0155
   LOCAL anode;                                                         0156
   REF anode;                                                           0157
   &anode ← [azsubn].dptrun;                                            0158
   WHILE &anode DO                                                      0159
      BEGIN                                                             0160
      ckshwcmd( &anode);                                                0161
      &anode ← anode.alternative;                                       0162
      END;                                                              0163
   RETURN;                                                              0164
   END.
                                                                        0165
   %%                                                                  0166
```

```
(nprint) PROCEDURE  % top level node interpretter %                    01037
   % FORMALS %                                                         01038
      (astr,    % adr of string to be appended to %                   01039
      anode,    % adr of tree to be interpretted %                    01040
      goingdown,        % if TRUE, look at alternatives to anode %    01041
      altsuc);          % if TRUE, adr of tree to continue with       01042
         after exhausting anode tree %                                01043
   % ALGORITHM %                                                       01044
      % this procedure calls itself recursively to follow the
      alternative paths of a node.  When it hits a node with no
      alternative it calls pnode to print this node and then it goes on
      the the succesor of this node.  If this node has no successor, it
      goes on to the tree passed to it as altsuc.  If it has someplace
      to go on to, it iterates to go on;  if it has no place to go on
      to, it outputs the current path and then returns. %              01045
   LOCAL svcsstkb;                                                     01046
   LOCAL STRING locstr[200];                                           01047
   REF astr, anode;                                                    01048
                                                                       01049
   svcsstkb ← csstkb;                                                  01050
   LOOP                                                                01051
      BEGIN                                                            01052
      IF inptrf THEN RETURN( FALSE );                                  01097
      IF goingdown AND anode.alternative AND                           01053
         (anode.alternative # &anode) THEN                             01054
            BEGIN  % recurse to get all alternatives %                 01055
            *locstr* ← *astr*;  % save current string %                01056
            goingdown ← nprint( &astr, anode.alternative, TRUE,
            altsuc);                                                   01057
            *astr* ← *locstr*;  % restore string %                     01058
            END                                                        01059
      ELSE                                                             01060
         IF (svcsstkb >= csstkx) OR (&anode = csstk[svcsstkb]) THEN
                                                                       01061
            BEGIN                                                      01062
            % print this node; if pnode returns TRUE then we are done
            we this part of the tree and need not go any further %     01063
               IF pnode( &astr, &anode, altsuc) THEN RETURN( FALSE );
                                                                       01064
            % go on to the succesor; if there isn't one then go on to
            altsuc and reset altsuc (so we don't loop infinitely) %
                                                                       01065
               &anode ← anode.nsuccessor;                              01066
               goingdown ← TRUE;                                       01067
               IF NOT &anode THEN &anode ← altsuc := FALSE;            01068
               BUMP svcsstkb;                                          01069
            % if real end of tree output this path & return, ELSE loop
            %                                                          01070
               IF (NOT &anode) THEN                                    01071
                  BEGIN                                                01072
                  onode( &astr );                                      01073
                  RETURN( FALSE );                                     01074
                  END;                                                 01075
            END                                                        01076
         ELSE RETURN( FALSE );                                         01077
      END;                                                             01078
```

END.

%%

01079
01080

```
(pnode) PROCEDURE  % print a node %                                    0814
   % FORMALS %                                                         0815
      (astr,    % adr of string to be appended to %                   0816
      anode,    % adr of tree to be interpretted %                    0817
      altsuc);          % if TRUE, adr of tree to continue with      0818
         after exhausting anode tree %                                0819
   % ALGORITHM %                                                       0820
      % this routine is responsible for printing (appending to the
      string astr) an individual node.  It is called from nprint, and
      returns FALSE if nprint is to go on to the succesor of this node.
       It returns TRUE if nprint is not to go on to the successor, i.e.
      we have taken responsibility for printing all paths from this
      node down (we do this by calling nprint recursively ourself). %
                                                                       0821
   LOCAL i, count, char, ptr;                                          0822
   LOCAL STRING locstr[200];                                           0823
   REF astr, anode;                                                    0824
                                                                       0825
   IF csstkb < csstkx THEN                                             0826
      IF &anode NOT= csstk[csstkb] THEN RETURN( TRUE )                 0827
      ELSE BUMP csstkb;                                                0828
   CASE anode.opcode OF                                                0829
      = $keyop:         % keyword %                                    0830
         % put keyword in string, first letter upper, rest lower %     0831
         BEGIN                                                         0832
         CASE cmdctl OF                                                0833
            = 0: NULL;          % want all commands %                  0834
            = 1:                % want DNLS commands %                 0835
               IF NOT anode.ctrl.dnlscmd THEN RETURN( TRUE );          0836
            = 2:                % want TNLS commands %                 0837
               IF NOT anode.ctrl.tnlscmd THEN RETURN( TRUE );          0838
         ENDCASE;                                                      0839
         IF NOT (anode.ctrl.dnlscmd AND anode.ctrl.tnlscmd) THEN       0840
            BEGIN                                                      0841
            IF anode.ctrl.dnlscmd AND NOT cmdctl THEN                  0842
               *astr* ← *astr*, "!DNLS!";                              0843
            IF anode.ctrl.tnlscmd AND NOT cmdctl THEN                  0844
               *astr* ← *astr*, "!TNLS!";                              0845
            END;                                                       0846
         char ← *[anode.addr]*[1];                                     0847
         IF char < 40B THEN                                            0848
            BEGIN                                                      0849
            ptr ← npstrad( char );                                     0850
            *astr* ← *astr*, *[ptr]*;                                  0851
            END                                                        0852
         ELSE *astr* ← *astr*, char;                                   0853
         count ← [anode.addr].L;                                       0854
         FOR i ← 2 UP UNTIL > count DO                                 0855
            BEGIN                                                      0856
            char ← *[anode.addr]*[i];                                  0857
            CASE char OF                                               0858
               IN ['A, 'Z]: char ← char .V 40B;                        0859
               ENDCASE;                                                0860
            *astr* ← *astr*, char;                                     0861
            END;                                                       0862
         *astr* ← *astr*, SP;                                          0863
```

```
    END;                                                          0864
= $confirm:         % confirmation %                              0865
    *astr* ← *astr*, "OK ";                                       0866
= $ssel:            % source selection %                          0867
    *astr* ← *astr*, "SOURCE ";                                   0868
= $dsel:            % destination selection %                     0869
    *astr* ← *astr*, "DESTINATION ";                              0870
= $lsel:            % literal selection %                         0871
    *astr* ← *astr*, "CONTENT ";                                  0872
= $vwspecs:         % viewspec selection %                        0873
    *astr* ← *astr*, "VIEWSPECS ";                                0874
= $levadj:          % level-adjust selection %                    0875
    *astr* ← *astr*, "LEVEL-ADJUST ";                             0876
= $necho, = $recho:       % noise words %                        0877
    *astr* ← *astr*, '(,  *(anode.addr)*, ") ";                   0878
= $option:          % CML option construct %                      0879
    BEGIN                                                         0880
    % place OPTION in string %                                    0881
       *astr* ← *astr*, "OPTION ";                                0882
    % print paths from here on down ourself, since this is now a
    tertiary tree.  when we finish the option tree, pick up with
    our own successor if there is one, otherwise pick up with
    altsuc %                                                      0883
       nprint( &astr, anode.addr, TRUE,                           0884
          IF anode.nsuccessor THEN anode.nsuccessor               0885
          ELSE altsuc );                                          0886
    % return TRUE since we have printed paths from here down %    0887
       RETURN( TRUE );                                            0888
    END;                                                          0889
= $pfcall:          % CML parsefunction call %                    0890
    % if this parsefunction does not appear in the table
    prsfunctions, then print nothing, and return false for normal
    processing;  if this parsefunction does appear in the table,
    the second word of its entry describes the action to be taken
    as follows:                                                   0891
       lefthalf of word = -1                                      0892
          call the procedure whose address is in the righthalf of
          the word, and pass the same arguments to this called
          procedure as were passed to us, and return the result of
          this called procedure.                                  0893
       lefthalf of word # -1                                      0894
          the righthalf of the word contains the address of a
          string to placed in the output string, and then return
          false for normal processing %                           0895
    CASE prsfunctions OF                                          0896
       <= 0:        % no entries in table %                       0897
          RETURN( FALSE );                                        0898
       ENDCASE                                                    0899
          BEGIN                                                   0900
          FOR i ← 0 UP UNTIL >= prsfunctions DO                   0901
             IF anode.addr = prsfunctions[2*i+1] THEN             0902
                CASE prsfunctions[2*i+2].LH OF                    0903
                   = 18M:           % lefthalf entry = -1 %       0904
                      % call the procedure, passing it proper
                      arguments and return the result of the
                      called procedure %                          0905
```

```
                        RETURN(                                          0906
                           [prsfunctions[2*i+2].RH]( &astr,
                           &anode, altsuc)   );                          0907
                     ENDCASE          % lefthalf entry # -1 %            0908
                        % append string from righthalf of word where
                        it belongs and return false %                    0909
                        BEGIN                                            0910
                        *astr* ← *astr*,                                 0911
                           *[prsfunctions[2*i+2].RH]*;                   0912
                        RETURN( FALSE );;                                0913
                        END;                                             0914
            % if entry not in table do nohing and return false %         0915
              IF i >= prsfunctions THEN RETURN( FALSE );                 0916
            END;                                                         0917
= $call:          % CML xroutine call %                                  0918
   % if this xroutine does not appear in the table xroutines,
   then print nothing, and return false for normal processing;
   if this xroutine does appear in the table, the second word of
   its entry describes the action to be taken as follows:                0919
      lefthalf of word = -1                                              0920
         call the procedure whose address is in the righthalf of
         the word, and pass the same arguments to this called
         procedure as were passed to us, and return the result of
         this called procedure.                                          0921
      lefthalf of word # -1                                              0922
         the righthalf of the word contains the address of a
         string to placed in the output string, and then return
         false for normal processing %                                   0923
   CASE xroutines OF                                                     0924
      <= 0:  % nothing in table %                                        0925
         RETURN( FALSE );                                                0926
      ENDCASE                                                            0927
         BEGIN                                                           0928
         FOR i ← 0 UP UNTIL >= xroutines DO                              0929
            IF anode.addr = xroutines[2*i+1] THEN                        0930
               CASE xroutines[2*i+2].LH OF                               0931
                  = 18M:          % lefthalf = -1 %                      0932
                     % call the procedure, passing it proper
                     arguments and return the result of the
                     called procedure %                                  0933
                        RETURN(                                          0934
                           [xroutines[2*i+2].RH]( &astr, &anode,
                           altsuc)   );                                  0935
                  ENDCASE          % lefthalf # -1 %                     0936
                     % append string from righthalf of word where
                     it belongs and return false %                       0937
                        BEGIN                                            0938
                        *astr* ← *astr*, *[xroutines[2*i+2].RH]*;
                                                                         0939
                        RETURN( FALSE );;                                0940
                        END;                                             0941
         % if xroutine not in table do nothing, return false %
                                                                         0942
           IF i >= xroutines THEN RETURN( FALSE );                       0943
         END;                                                            0944
= $execute:       % CML rule construct %                                0945
```

```
% if this rule does not appear in the table prules, then do
the following:                                                  0946
   if there are more than nmalt paths through the rule then
   place the name of the rule in the output string and return
   false;                                                       0947
      (if entire rule is optional make a pass putting out path
      without the rule appearing)                               0948
   if there are nmalt paths or less through the rule, then
   call nprint recursively to expand the rule.                  0949
      if the entire rule is optional then return false so we
      will make a path that doesn't include the rule,
      otherwise return true                                     0950
if this rule does appear in the table, the second word of its
entry describes the action to be taken as follows:              0951
   lefthalf of word = -1                                        0952
      call the procedure whose address is in the righthalf of
      the word, and pass the same arguments to this called
      procedure as were passed to us, and return the result of
      this called procedure.                                    0953
   lefthalf of word # -1                                        0954
      the righthalf of the word contains the address of a
      string to placed in the output string, and then return
      false for normal processing                               0955
         (if entire rule is optional make a pass putting out
         path without the rule appearing)                       0956
if this rule is not expanded either because it had too many
paths or because a string replacement entry existed in the
prules table, then an entry is made in the post-processing
stack  (see appstk for details of this stack ) %                0957
CASE csstkb OF                                                  0958
   >= csstkx:                                                   0959
      BEGIN                                                     0960
      IF anode.addr THEN                                        0961
         CASE prules OF                                         0962
            <= 0:     % no entries in table %                   0963
               IF ckaltcnt( anode.addr ) > nmalt THEN           0964
                  BEGIN  % too many paths %                     0965
                  IF ckoptnode( anode.addr ) THEN               0966
                     BEGIN  % entire rule optional %            0967
                     *locstr* ← *astr*;                         0968
                     nprint( &astr, IF anode.nsuccessor THEN
                     anode.nsuccessor ELSE altsuc, TRUE,
                     FALSE);                                    0969
                     *astr* ← *locstr*;                         0970
                     END;                                       0971
                  % get rule name to output string %            0972
                     gtruln( anode.addr, $locstr);              0973
                     *astr* ← *astr*, *locstr*, SP;             0974
                  % make entry in post-processing stack %       0975
                     appstk( anode.addr"@   ]:
                  return[ false ]:
                  end
               else
                  begin  % expand rule: recursive call to
                  nprint %                                      0980
                  *locstr* ← *astr*;                            0981
```

```
                    nprint( &astr, anode.addr, TRUE,          0982
                       IF anode.nsuccessor THEN anode.nsuccessor
                                                              0983
                       ELSE altsuc);                          0984
                    *astr* ← *locstr*;                        0985
                    % if entire rule optional, return false %
                                                              0986
                       IF ckoptnode( anode.addr ) THEN        0987
                          RETURN( FALSE );                    0988
                    END;                                      0989
              ENDCASE                                         0990
                 BEGIN                                        0991
                 FOR i ← 0 UP UNTIL >= prules DO              0992
                    IF anode.addr = prules[2*i+1] THEN        0993
                    CASE prules[2*i+2].LH OF                  0994
                       = 18M:        % lefthalf = -1 %       0995
                          % call the procedure, passing it
                          proper arguments and return the
                          result of the called procedure %
                                                              0996
                             RETURN(                          0997
                                [prules[2*i+2].RH]( &astr,
                                &anode, altsuc)   );          0998
                          ENDCASE       % lefthalf # -1 %     0999
                             % append string from righthalf of
                             word where it belongs and return
                             false %                          01000
                                BEGIN                         01001
                                IF ckoptnode( anodg.addr ) THEN
                                                              01002
                                   % if entire rule optional,
                                   make pass without rule %   01003
                                      BEGIN                   01004
                                      *locstr* ← *astr*;      01005
                                      nprint(                 01006
                                      &astr, IF anode.nsuccessor
                                      THEN anode.nsuccessor ELSE
                                      altsuc, TRUE, FALSE);   01007
                                      *astr* ← *locstr*;      01008
                                      END;                    01009
                                *astr* ← *astr*,              01010
                                   *[prules[2*i+2].RH]*;      01011
                                % make entry in post-processing
                                stack %                       01012
                                   appstk( anode.addr,        01013
                                      prules[2*i+2].RH );     01014
                                RETURN( FALSE );              01015
                                END;                          01016
                 % if rule not in table, pretend table had no
                 entries %                                    01017
                    IF i >= prules THEN REPEAT CASE( 0 );     01018
                 END;                                         01019
   RETURN( TRUE );                                            01020
   END;                                                       01021
ENDCASE                                                       01022
   BEGIN  % expand rule: recursive call to nprint %           01023
```

```
            *locstr* ← *astr*;                                          01024
            nprint( &astr, anode.addr, TRUE,                            01025
               IF anode.nsuccessor THEN anode.nsuccessor                01026
               ELSE altsuc);                                            01027
            *astr* ← *locstr*;                                          01028
            % if entire rule optional, return false %                   01029
               IF ckoptnode( anode.addr ) THEN                          01030
                  RETURN( FALSE );                                      01031
            RETURN( TRUE );                                             01098
            END;                                                        01032
   ENDCASE;                                                             01033
RETURN( FALSE );                                                        01034
END.

%%                                                                      01035
                                                                        01036
```

```
(onode) PROCEDURE       % output a path %                                       0411
  % FORMALS %                                                                   0412
     ( astr );          % adr string to be output %                            0413
  % ALGORITHM %                                                                 0414
     % this procedure will output a path.  the method of outputting is
     controlled by the value of omode as follows:                              0415
        0 - do typeas                                                          0416
        1 - do insert statement                                                0417
           cinstid has stid of stmnt to insert after                           0418
           ilevel has level adjust value                                       0419
        2 - do send using cinstid as adr of seq work area                      0420
        3 - call fbctl to display path %                                       0421
  LOCAL TEXT POINTER tp1, tp2;                                                 0422
  REF astr;                                                                    0423
  CASE omode OF                                                                0424
     = 0:                                                                      0425
        BEGIN                                                                  0426
        *astr* ← EOL, *astr*;                                                  0427
        typeas( &astr );                                                       0428
        END;                                                                   0429
     = 1:                                                                      0430
        BEGIN                                                                  0431
        FIND SF(*astr*) ↑tp1 SE(*astr*) ↑tp2;                                  0432
        cinstid ← cinssta(cinstid, ilevel := levsuc, $tp1, $tp2);              0433
        END;                                                                   0434
     = 2:                                                                      0435
        BEGIN                                                                  0436
        send( cinstid, &astr);                                                 0437
        END;                                                                   0438
     = 3:                                                                      0439
        BEGIN                                                                  0440
        *astr* ← EOL, EOL, *astr*;                                             0441
        fbctl( fbaddlit, &astr);                                               0442
        END;                                                                   0443
    ENDCASE REPEAT CASE( 0 );                                                  0444
  RETURN;                                                                      0445
  END.

                                                                               0446
  %%                                                                          0447
```

```
(ckoptnode) PROCEDURE  % check if entire rule is optional %                    0448
   % FORMALS %                                                                 0449
      ( &node );                                                               0450
   % ALGORITHM %                                                               0451
      % returns true if entire rule is optional (i.e., there are no
      non-optional paths through rule)                                         0452
      this procedure can call itself recursively if it encounters a
      rule within a rule.                                                      0453
      it returns false as soon as it finds a non-optional path.  if it
      finds no such path, it returns true %                                    0454
   REF node;                                                                   0455
                                                                               0456
   CASE node.opcode OF                                                         0457
      = $option:                                                               0458
         IF &node # node.alternative THEN                                      0459
            IF &node ← node.alternative THEN REPEAT CASE                       0460
            ELSE RETURN( TRUE )                                                0461
         ELSE RETURN( TRUE );                                                  0462
      = $xecute:                                                               0463
         IF ckoptnode( node.addr ) THEN                                        0464
            IF &node # node.alternative THEN                                   0465
               IF &node ← node.alternative THEN REPEAT CASE                    0466
               ELSE RETURN( TRUE )                                             0467
            ELSE RETURN( TRUE )                                                0468
         ELSE RETURN( FALSE );                                                 0469
      ENDCASE RETURN( FALSE );                                                 0470
   END.                                                                        0471
                                                                               0472
   %%
```

```
(ckaltcnt) PROCEDURE  % count number of paths through rule %          0473
   % FORMALS %                                                        0474
      ( anode );                                                      0475
   % ALGORITHM %                                                     0476
      % returns number of alternative paths through rule  without
      expanding any rules that appear and without checking to see if
      embedded rules are optional                                     0477
      this routine calls itself recursively for successors and
      iterates, while counting, for alternatives %                    0478
   LOCAL i;                                                           0479
   REF anode;                                                         0480
                                                                      0481
   i ← 0;                                                             0482
   WHILE &anode DO                                                    0483
      BEGIN                                                           0484
      IF anode.nsuccessor THEN                                        0485
         i ← i - 1 + ckaltcnt( anode.nsuccessor );                    0486
            % subtract one so we dcnt count this node twice %         0487
      BUMP i;                                                         0488
      IF &anode = anode.alternative THEN EXIT LOOP;                   0489
      &anode ← anode.alternative;                                     0490
      END;                                                            0491
   RETURN( i );                                                       0492
   END.
                                                                      0493
   %%                                                                 0494
```

```
(gtruln) PROCEDURE  % get name of rule %                                    0495
   % FORMALS %                                                              0496
      (raddr,  % adr rule whose name is to be gotten %                      0497
      astr );  % adr string to get rule name %                              0498
   % ALGORITHM %                                                            0499
      % uses ddgtsymbol to get name of rule into string and will remove
      any block name from the front for local symbols %                     0500
   LOCAL TEXT POINTER tpl;                                                  0501
   REF astr;                                                                0502
                                                                            0503
   ddgtsymbol( &astr, raddr);                                               0504
   IF FIND SF(*astr*) '[ [']] ↑tpl THEN                                     0505
      *astr* ← tpl SE(*astr*);                                              0506
   RETURN;                                                                  0507
   END.
                                                                            0508
   %%                                                                      0509
```

```
(appstk) PROCEDURE  % make an entry in the post-processing stack %  0510
   % FORMALS %                                                      0511
      (raddr,  % address of rule to be added to stack %             0512
      astr);   % adr of string from prules table or zero %         0513
   % ALGORITHM %                                                   0514
      % if the passed rule is not already in the post-processing stack
      then it is added to the stack                                 0515
      if the rule address appears in the table noexpand, then no entry
      is made in the table                                          0516
       each entry in the stack has the rule address in the right half
      and the left half has the string address (from prules table) or
      zero in it %                                                  0517
   LOCAL i;                                                         0518
                                                                    0519
   % return if entry already exists %                               0520
      FOR i ← 0 UP UNTIL >= pstkp DO                                0521
         IF pstk[i+1].RH = raddr THEN RETURN;                       0522
   % dont make entry if raddr is in table noexpand %                0523
      FOR i ← 1 UP UNTIL > noexpand DO                              0524
         IF noexpand[i].RH = raddr THEN RETURN;                     0525
   % make entry %                                                   0526
      pstk[pstkp ← pstkp+1] ← raddr;                                0527
      pstk[pstkp].LH ← astr;                                        0528
   RETURN;                                                          0529
   END.
                                                                    0530
   %%                                                              0531
```

```
% special rule, parsefunction, xroutine handlers %                          0532
  (prconfirmlook) PROCEDURE  % lookconfirm parsefunction handler %          0533
    % FORMALS %                                                             0534
      (astr,          % adr of string to be appended to %                  0535
      anode,          % adr of tree to be interpretted %                   0536
      altsuc);        % if TRUE, adr of tree to continue with              0537
         after exhausting anode tree %                                      0538
    % ALGORITHM %                                                           0539
      % if the next non-trivial successor node in the tree is a
      confirm node then do nothing and return false for normal
      processing                                                            0540
        if the next non-trivial successor node in the tree is a
      keyop, vwspecs, levadj, option, prcall, xecute, or call node
      then place the string OK in the output string and return false
      for normal processing                                                 0541
        if the next non-trivial successor node in the tree is a
      lsel, dsel, or ssel node then place the string BUG in the
      output string                                                         0542
           if this non-trivial node has no alternatives then call
        nprint recursively to continue since we are skipping some
        nodes in the tree and then return true                              0543
           if this non-trivial node has alternatives then call
        nprint recursively to continue since we are skipping some
        nodes in the tree and then call nprint recursively again to
        get the alternatives and then return true                           0544
        if the next node is a trivial node without alternatives then
      loop                                                                  0545
        if the next node is a trivial node with alternatives then
      place the string OK in the output string and return false for
      normal processing %                                                   0546
    LOCAL tsuc, talt;                                                       0547
    LOCAL STRING locstr[200];                                               0548
    REF astr, anode;                                                        0549
                                                                            0550
    CASE [anode.nsuccessor].opcode OF                                       0551
       = $confirm: RETURN( FALSE );                                         0552
       =$keyop, =$vwspecs, =$levadj, =$option, =$prcall, =$xecute,
       =$call:                                                              0553
          BEGIN                                                             0554
          *astr* ← *astr*, "OK ";                                           0555
          RETURN( FALSE );                                                  0556
          END;                                                              0557
       = $ssel, = $lsel, = $dsel:                                           0558
          CASE [anode.nsuccessor].alternative OF                            0559
             = 0, = &anode:                                                 0560
                BEGIN                                                       0561
                *astr* ← *astr*, "BUG ";                                    0562
                &anode ← anode.nsuccessor;                                  0563
                nprint(                                                     0564
                   &astr,                                                   0565
                   IF anode.nsuccessor                                      0566
                      THEN anode.nsuccessor                                 0567
                      ELSE altsuc,                                          0568
                   TRUE,                                                    0569
                   IF anode.nsuccessor THEN altsuc ELSE FALSE               0570
                   );                                                       0571
```

```
            RETURN( TRUE );                                          0572
            END;                                                     0573
         ENDCASE                                                     0574
            BEGIN                                                    0575
            *locstr* ← *astr*;                                       0576
            *astr* ← *astr*, "BUG ";                                 0577
            IF [anode.nsuccessor].nsuccessor THEN                    0578
               BEGIN                                                 0579
               tsuc ← [anode.nsuccessor].nsuccessor;                 0580
               talt ← altsuc;                                        0581
               END                                                   0582
            ELSE                                                     0583
               IF altsuc THEN                                        0584
                  BEGIN                                              0585
                  tsuc ← altsuc;                                     0586
                  talt ← 0;                                          0587
                  END                                                0588
               ELSE tsuc ← FALSE;                                    0589
            IF tsuc THEN nprint( &astr, tsuc, TRUE, talt)            0590
            ELSE onode( &astr );                                     0591
            *astr* ← *locstr*, "OK ";                                0592
            nprint(                                                  0593
               &astr, [anode.nsuccessor].alternative, TRUE,
               altsuc);                                              0594
            RETURN( TRUE );                                          0595
            END;                                                     0596
   ENDCASE                                                           0597
      CASE [anode.nsuccessor].alternative OF                         0598
         = 0, = anode.nsuccessor:                                    0599
            BEGIN                                                    0600
            &anode ← anode.nsuccessor;                               0601
            REPEAT CASE 2;                                           0602
            END;                                                     0603
         ENDCASE                                                     0604
            BEGIN                                                    0605
            *astr* ← *astr*, "OK ";                                  0606
            RETURN( FALSE );                                         0607
            END;                                                     0608
END.
                                                                     0609
%%                                                                   0610
```

```
(pbuglook) PROCEDURE  % lookbug parsefunction handler %                    0611
   % FORMALS %                                                             0612
      (astr,          % adr of string to be appended to %                  0613
      anode,          % adr of tree to be interpretted %                   0614
      altsuc);        % if TRUE, adr of tree to continue with              0615
         after exhausting anode tree %                                     0616
   % ALGORITHM %                                                           0617
      % if the next non-trivial successor node in the tree is a
      confirm node then do nothing and return false for normal
      processing                                                           0618
        if the next non-trivial successor node in the tree is a
      keyop, vwspecs, levadj, option, pfcall, xecute, or call node
      then place the string BUG in the output string and return
      false for normal processing                                          0619
        if the next non-trivial successor node in the tree is a
      lsel, dsel, or ssel node then place the string BUG in the
      output string                                                        0620
           if this non-trivial node has no alternatives then call
         nprint recursively to continue since we are skipping some
         nodes in the tree and then return true                            0621
           if this non-trivial node has alternatives then call
         nprint recursively to continue since we are skipping some
         nodes in the tree and then call nprint recursively again to
         get the alternatives and then return true                         0622
        if the next node is a trivial node without alternatives then
      loop                                                                 0623
        if the next node is a trivial node with alternatives then
      place the string OK in the output string and return false for
      normal processing %                                                  0624
   LOCAL tsuc, talt;                                                       0625
   LOCAL STRING locstr[200];                                               0626
   REF astr, anode;                                                        0627
                                                                           0628
   CASE [anode.nsuccessor].opcode OF                                       0629
      = $confirm: RETURN( FALSE );                                         0630
      =$keyop, =$vwspecs, =$levadj, =$option, =$pfcall, =$xecute,
      =$call:                                                              0631
         BEGIN                                                             0632
         *astr* ← *astr*, "BUG ";                                          0633
         RETURN( FALSE );                                                  0634
         END;                                                              0635
      = $ssel, = $lsel, = $dsel:                                           0636
         CASE [anode.nsuccessor].alternative OF                            0637
            = 0, = &anode:                                                 0638
               BEGIN                                                       0639
               *astr* ← *astr*, "BUG ";                                    0640
               &anode ← anode.nsuccessor;                                  0641
               nprint(                                                     0642
                  *astr,                                                   0643
                  IF anode.nsuccessor                                      0644
                     THEN anode.nsuccessor                                 0645
                     ELSE altsuc,                                          0646
                  TRUE,                                                    0647
                  IF anode.nsuccessor THEN altsuc ELSE FALSE               0648
                  );                                                       0649
               RETURN( TRUE );                                             0650
```

```
            END;                                                    0651
         ENDCASE                                                    0652
            BEGIN                                                   0653
            *locstr* ← *astr*;                                      0654
            *astr* ← *astr*, "BUG ";                                0655
            IF (anode.nsuccessor).nsuccessor THEN                   0656
               BEGIN                                                0657
               tsuc ← (anode.nsuccessor).nsuccessor;                0658
               talt ← altsuc;                                       0659
               END                                                  0660
            ELSE                                                    0661
               IF altsuc THEN                                       0662
                  BEGIN                                             0663
                  tsuc ← altsuc;                                    0664
                  talt ← 0;                                         0665
                  END                                               0666
               ELSE tsuc ← FALSE;                                   0667
            IF tsuc THEN nprint( &astr, tsuc, TRUE, talt)           0668
            ELSE onode( &astr );                                    0669
            *astr* ← *locstr*, "BUG ";                              0670
            nprint(                                                 0671
               &astr, (anode.nsuccessor).alternative, TRUE,
               altsuc);                                             0672
            RETURN( TRUE );                                         0673
            END;                                                    0674
   ENDCASE                                                          0675
      CASE (anode.nsuccessor).alternative OF                        0676
         = 0, = anode.nsuccessor:                                   0677
            BEGIN                                                   0678
            &ancde ← anode.nsuccessor;                              0679
            REPEAT CASE 2;                                          0680
            END;                                                    0681
         ENDCASE                                                    0682
            BEGIN                                                   0683
            *astr* ← *astr*, "BUG ";                                0684
            RETURN( FALSE );                                        0685
            END;                                                    0686
END.                                                                0687

%%                                                                  0688
```

```
(prnumlook) PROCEDURE  % looknum parsefunction handler %                    0689
   % FORMALS %                                                              0690
      (astr,          % adr of string to be appended to %                   0691
      anode,          % adr of tree to be interpretted %                    0692
      altsuc);        % if TRUE, adr of tree to continue with               0693
         after exhausting anode tree %                                      0694
   % ALGORITHM %                                                            0695
      % if the next non-trivial successor node in the tree is a
      confirm, keyop, vwspecs, levadj, option, pfcall, xecute, or
      call node then place the string NUMBER in the output string
      and return false for normal processing                                0696
        if the next non-trivial successor node in the tree is a
      lsel, dsel, or ssel node then place the string NUMBER in the
      output string                                                         0697
           if this non-trivial node has no alternatives then call
        nprint recursively to continue since we are skipping some
        nodes in the tree and then return true                              0698
           if this non-trivial node has alternatives then call
        nprint recursively to continue since we are skipping some
        nodes in the tree and then call nprint recursively again to
        get the alternatives and then return true                           0699
        if the next node is a trivial node without alternatives then
      loop                                                                  0700
        if the next node is a trivial node with alternatives then
      place the string OK in the output string and return false for
      normal processing %                                                   0701
   LOCAL tsuc, talt;                                                        0702
   LOCAL STRING locstr[200];                                                0703
   REF astr, anode;                                                         0704
                                                                            0705
   CASE [anode.nsuccessor].opcode OF                                        0706
      =$confirm, =$keyop, =$vwspecs, =$levadj, =$option, =$pfcall,
      =$xecute, =$call:                                                     0707
         BEGIN                                                              0708
         *astr* ← *astr*, "NUMBER ";                                        0709
         RETURN( FALSE );                                                   0710
         END;                                                               0711
      = $ssel, = $lsel, = $dsel:                                            0712
         CASE [anode.nsuccessor].alternative OF                             0713
            = 0, = &anode:                                                  0714
               BEGIN                                                        0715
               *astr* ← *astr*, "NUMBER ";                                  0716
               &anode ← anode.nsuccessor;                                   0717
               nprint(                                                      0718
                  &astr,                                                    0719
                  IF anode.nsuccessor                                       0720
                     THEN anode.nsuccessor                                  0721
                     ELSE altsuc,                                           0722
                  TRUE,                                                     0723
                  IF anode.nsuccessor THEN altsuc ELSE FALSE                0724
                  );                                                        0725
               RETURN( TRUE );                                              0726
               END;                                                         0727
            ENDCASE                                                         0728
               BEGIN                                                        0729
               *locstr* ← *astr*;                                           0730
```

```
            *astr* ← *astr*, "NUMBER ";                                  0731
            IF (anode.nsuccessor).nsuccessor THEN                        0732
               BEGIN                                                     0733
               tsuc ← (anode.nsuccessor).nsuccessor;                     0734
               talt ← altsuc;                                            0735
               END                                                       0736
            ELSE                                                         0737
               IF altsuc THEN                                            0738
                  BEGIN                                                  0739
                  tsuc ← altsuc;                                         0740
                  talt ← 0;                                              0741
                  END                                                    0742
               ELSE tsuc ← FALSE;                                        0743
            IF tsuc THEN nprint( &astr, tsuc, TRUE, talt)                0744
            ELSE oncde( &astr );                                         0745
            *astr* ← *locstr*, "NUMBER ";                                0746
            nprint(                                                      0747
               &astr, (anode.nsuccessor).alternative, TRUE,
               altsuc);                                                  0748
            RETURN( TRUE );                                              0749
            END;                                                         0750
      ENDCASE                                                            0751
         CASE (anode.nsuccessor).alternative OF                          0752
            = 0, = anode.nsuccessor:                                     0753
               BEGIN                                                     0754
               &anode ← anode.nsuccessor;                                0755
               REPEAT CASE 2;                                            0756
               END;                                                      0757
            ENDCASE                                                      0758
               BEGIN                                                     0759
               *astr* ← *astr*, "NUMBER ";                               0760
               RETURN( FALSE );                                          0761
               END;                                                      0762
END.
                                                                         0763
%%                                                                      0764
```

```
(prhchk) PROCEDURE  % hchk rule handler %                                0765
   % FORMALS %                                                           0766
      (astr,        % adr of string to be appended to %                 0767
      anode,        % adr of tree to be interpretted %                  0768
      altsuc);      % if TRUE, adr of tree to continue with             0769
         after exhausting anode tree %                                  0770
   % ALGORITHM %                                                         0771
      % since the rule hchk is a recurive rule in the grammar this
      procedure checks to make sure we dont loop indefinetly by
      checking to see if we appear on the return stack %                0772
   LOCAL frameptr;                                                      0773
   LOCAL STRING locstr[200];                                            0774
   REF astr, anode;                                                     0775
                                                                        0776
   frameptr ← m.RH;                                                     0777
   WHILE (frameptr ← frameptr.RH) > $gstack DO                          0778
      IF [frameptr].RH IN [$prhchk, $eprhchk] THEN RETURN( TRUE )
                                                                        0779
      ELSE frameptr ← [frameptr-1];                                     0780
   % haven't been here yet so expand the rule %                         0781
      IF ckaltcnt( anode.addr ) > nmalt THEN                            0782
         BEGIN  % too many paths %                                      0783
         IF ckoptnode( anode.addr ) THEN                                0784
            BEGIN  % entire rule optional %                             0785
            *locstr* ← *astr*;                                          0786
            nprint( &astr, IF anode.nsuccessor THEN anode.nsuccessor
            ELSE altsuc, TRUE, FALSE);                                  0787
            *astr* ← *locstr*;                                          0788
            END;                                                        0789
         % get rule name to output string %                             0790
            gtruln( anode.addr, $locstr);                               0791
            *astr* ← *astr*, *locstr*, SP;                              0792
         % make entry in post-processing stack %                        0793
            appstk( anode.addr, 0 );                                    0794
         RETURN( FALSE );                                               0795
         END                                                            0796
      ELSE                                                              0797
         BEGIN  % expand rule: recursive call to nprint %               0798
         *locstr* ← *astr*;                                             0799
         nprint( &astr, anode.addr, TRUE,                               0800
            IF anode.nsuccessor THEN anode.nsuccessor                   0801
            ELSE altsuc);                                               0802
         *astr* ← *locstr*;                                             0803
         % if entire rule optional, return false %                      0804
            IF ckoptnode( anode.addr ) THEN                             0805
               RETURN( FALSE );                                         0806
         END;                                                           0807
   RETURN( TRUE );                                                      0808
   (eprhchk):                                                           0809
   END.
                                                                        0810
   %%                                                                  0811
```

```
(prjmpcmd) PROCEDURE  % force expansion of rule JMPCOMS %          01081
   % FORMALS %                                                     01092
      (astr,          % adr of string to be appended to %        01093
      anode,          % adr of tree to be interpretted %         01094
      altsuc);        % if TRUE, adr of tree to continue with    01095
         after exhausting anode tree %                           01096
   LOCAL STRING locstr[200];                                      01082
   REF astr, anode;                                               01083
   *locstr* ← *astr*;                                             01084
   nprint( &astr, anode.addr, TRUE,                               01085
      IF anode.nsuccessor THEN anode.nsuccessor                   01086
      ELSE altsuc);                                               01087
   *astr* ← *locstr*;                                             01088
   RETURN( TRUE );                                                01089
   END.
                                                                  01090
   %%                                                            01091
```

UDPGEN.

```
< NLS. DSPGEN.NLS:396, >, 15-OCT-74 09:14 KJM ;;;;;
FILE dspgen  %  L10  <rel-nls>dspgen  %% (L10,) (rel-nls,dspgen.rel,) %
                                                                      02
%.....Declarations.....%                                              03
   REF msgda, litda, namda, cflda, ltvsda, vspcda, rt, art, mkrptr;   04
   REF lppch; % Line processor printer char routine %              02582
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4, s = 9, m = 10;            05
   DECLARE STRING                                                     06
      mt = "Empty",                                                   07
         %string can be changed withouut program modification         08
         if less than 20 characters in length%                        09
      dotstr = ".............................",
      unknown = "<UKC>" %unknown char detected in npstrad%;           010
   DECLARE EXTERNAL deltadata = 1, hazeltine = 2;                     011
   DECLARE  %source codes for LSRT%                                   012
      srcnul = 0, srcstat = 1, srcstno = 2, srcsig = 3, srcast        013
      = 4, srcdot = 5;                                                014
   DECLARE shrink = 4; %how many consecutive times user can           015
      have too much LSRT before system reallocates%                   016
   DECLARE  %display commands%                                        017
      writelsg = 1, movelsg = 2, deletlsg = 3, suppressda = 4,
      restoreda = 5, blankda = 6;                                     018
   SET BIT = 2518;                                                    019
                                                                      020
   DECLARE  %command processing constants%                            021
      forward = 1, backward = 2;                                      022
%.....CORE-NLS.....%
                                                                      023
   %....Display format control....%                                   024
      (dafrmt)   PROCEDURE (da, oldsw); %display area formatter%      025
         %This routine generates the display image for a text display
         area, assuming the universal display.  It is driven entirely
         by the information available in the display area record (da).
         Returns the t-pointer for the next statement character after
         the last one which was displayed.  oldsw is the address of an
         existing sequence work area or zero.  If zero, then a sequence
         is opened and closed in this procedure.%                     026
         %----------------%                                           027
         LOCAL end, stid, charct, sa, ptr, fpgindx, frzindx, endflg,
         fmtstd, sv1, sv2, frzarray[40];                              028
         REF da, sa, oldsw;                                           029
         IF NOT da.daexis THEN err(0"Fatal display error in DAFRMT");
                                                                      030
         IF NOT da.davspec.vsdaft THEN                                031
            BEGIN %dafrmt viewspec not set%                           032
            IF lplitreset THEN                                     05789
               BEGIN                                               05790
               litline ← lplitline;                                05791
               litreset ← FALSE;                                   05792
               litapflag ← TRUE;                                   05793
               rstlit();                                           05794
               lplitreset ← FALSE;                                 05795
               END;                                                05796
            IF NOT litreset THEN rstlit();                            033
            IF bmcnt THEN bmoff();                                    034
            RETURN(endfil, 1, TRUE);                                  035
```

```
   END;                                                          036
%..INITIALIZATION....%                                           037
   %..update previous file in da - for jump routines....%        038
      da.dapstf ← da.dacsp.stfile;                               039
   lplitreset ← FALSE;                                          05797
   %..if csp is not valid, set to origin..%                      040
      IF NOT da.daempty AND                                      041
      NOT goodrng(da.dacsp) THEN                                 042
         BEGIN                                                   043
         da.dacsp.stpsid ← origin;                               044
         da.daccnt ← 1;                                          045
         END;                                                    046
   %initialize frzarray pointer%                                 047
      ptr ← 0;                                                   048
   %initialize first commands block to empty%                    049
      cmdinit($commands, &da);                                   050
   %..initialize marker flag to false..%                         051
      mkrflg ← FALSE;                                            052
   %..initialize vertical beam position..%                       053
      da.dacrow ← 0;                                             054
   IF NOT litreset THEN rstlit();                                055
   IF bmcnt THEN bmoff();                                        056
   clearda(&da);%clear the display image %                       057
   clrall(&da, TRUE); % zero LSRT entries %                      058
%..Display Generation..%                                         059
   %..initialize LSRT (line segment reference table)
   pointer..%                                                    060
      rttop ← &rt ← da.dalsrt; %start of LSRT%                   061
      rte ← da.dalsz * lsrtl + &rt; %end of LSRT%                062
      rtsize ← da.dalsz;                                         063
   IF da.daempty OR da.dacsp = endfil THEN                       064
      BEGIN                                                      065
      damt(&da);                                                 066
      RETURN(endfil, 1, TRUE);                                   067
      END;                                                       068
   %set values to zero so system will blow up if they are
   used%                                                         069
      da.dalsrt ← da.dalsz ← 0;                                  070
   frzfrmt(&da);                                                 071
   &sa ←  IF &oldsw THEN &oldsw                                  072
      ELSE openseq (fmtstd ← da.dacsp, seqend(fmtstd, sv1 ←
      da.davspec, sv2 ← da.davspc2), sv1, sv2, da.dausqcod,
      da.dacacode);                                              073
   charct ← da.daccnt;                                           074
   UNTIL da.dacrow > da.damrow OR &rt >= rte                     075
   OR (stid ← seqgen (&sa)) = endfil DO                          076
      BEGIN                                                      077
      strfrmt (&da, stid, charct, sa.swclvl, sa.swslvl,
      sa.swsvw, sa.swvspec, sa.swvsp2);                          078
      charct ← 1;                                                079
      IF NOT stid.stastr THEN                                    080
         BEGIN %freeze the page in core if not already frozen%
                                                                 081
         fpgindx ← lodsdb(getsdb(stid));                         082
         %check if this page already frozen%                     083
         FOR frzindx ← 0 UP 1 UNTIL >= ptr DO                    084
```

```
            IF frzarray[frzindx] = fpgindx THEN EXIT;          085
         IF frzindx >= ptr THEN %not previously frozen by
         dafrmt%                                               086
            frzblk(frzarray[ptr := ptr + 1] ← fpgindx, 1);     087
         END;                                                  088
      IF nldevice = devlproc THEN %temporary patch for LP%     089
         BEGIN                                                 090
         prcmds(&da);                                          091
         cmdinit($commands, &da);                              092
         END;                                                  093
      END;                                                     094
   prcmds(&da); %process the accumulated display JSYS
   commands%                                                   095
   % If endflg is FALSE, we haven't reached the end of the
   material to be displayed  befoe the end of the screen. %  096
   endflg ←                                                    097
      IF stid # endfil AND da.dacrow > da.damrow THEN FALSE    098
      ELSE TRUE;                                               099
   da.davspec ← sa.swvspec;                                   0100
   da.davspc2 ← sa.swvsp2;                                    0101
   dspvsp(da.davspec, da.davspc2, 3);                         0102
   stid ← dgstid;  %dgstid, dgstml, dgstl set by stfrmt%      0103
   charct ← dgstl;                                            0104
%..termination..%                                             0105
   UNTIL (ptr ← ptr - 1) < 0 DO                               0106
      frzblk(frzarray[ptr], -1);                              0107
   %see if screen is really empty - could be i-viewspec didn't
   pass any statements%                                       0108
      da.dalsrt ← &rt; %starting address for damt to format
      into%                                                   0109
      IF da.dacrow = 0  OR                                    0110
      (da.dacrow = da.davinc AND                              0111
      [da.dalsrt].rtsrce = srcdot) THEN                       0112
         BEGIN                                                0113
         damt(&da);                                           0114
         stid ← endfil;                                       0115
         charct ← 1;                                          0116
         END;                                                 0117
   IF NOT &oldsw THEN closeseq (&sa);                         0118
%restore da LSRT table address, size %                        0119
   da.dalsrt ← rttop; %it may have changed%                   0120
   da.dalsz ← rtsize;                                         0121
RETURN(stid, charct, endflg);                                 0122
END.                                                          0123
                                                              0124
(daupdate) PROCEDURE (da); %update display area image%       05491
   %This routine updates the display image in the given text
   display area (da) by reformating those statements which have
   changed.%                                                  05492
   %Wherever possible, statements on the current screen are
   copied over to the new screen.  Globals cdtype, cdstd1, cdstd2
   and cdstop describe the kind of change that has taken place in
   the file.%                                                 05493
   %the reason for all the messing around with table pointers is
   that if stfrmt gets called, the new table being fomatted can
   change physical locations and size (more than once)
```

```
NOTE:  assumptions are made that imply that only statements on
the screen have been edited.%                                          05494
%-----------------%                                                    05495
                                                                       05496
LOCAL
   deleted,     %flag used to see what's missing from new
   table%                                                              05497
   ls,          %current line segment entry%                          05498
   ols,         %old line segment entry%                              05499
   nls,         %next line segment entry%                             05500
   origls,      %original line segment entry%                         05501
   sa,          %sequence work area%                                  05502
   stid,        %current STID being considered for formatting%
                                                                       05503
   bp,   %byte pointer%                                               05504
   length,                                                             05505
   endstd,         % End stid in sequence to be formatted. %
                                                                       05506
   svstd,                                                              05507
   sv1,     %Viewspec word 1 saved from first sequence work
   area. %                                                             05508
   sv2,     %Viewspec word 2 saved from first sequence work
   area. %                                                             05509
   slvl,        %base level saved from first wequence work
   area%                                                               05510
   match,       %flag: whether srch found the STID in old
   table%                                                              05511
   litup,       %flag: TRUE if lit was being displayed%               05512
   nextsup,     %addr of next ls to suppress for literal
   feedback:  for line prccessor%                                      05513
   copyflag,    %TRUE if lsrt entries copied from old to new
   lsrt%                                                               05514
   newtop,      %new logical top to minimize searching old
   table%                                                              05515
   tabltop,     %start of search in old table%                        05516
   newaulsrt,   %temp for new aux table allocation%                   05517
   rtsave;      %da.dalsrt on entry%                                  05518
REF nls, da, ls, ols, origls, sa, rt, art, mkrptr;                     05519
                                                                       05520
IF NOT da.daexis THEN err($"Fatal display error in DAUPDATE");
                                                                       05521
IF bmcnt THEN bmoff();                                                 05522
IF NOT da.davspec.vsdaft THEN                                          05523
   BEGIN %defer recreate display requested%                            05524
   IF lplitreset THEN                                                  05780
      BEGIN                                                            05781
      litline ← lplitline;                                             05782
      litreset ← FALSE;                                                05783
      litapflag ← TRUE;                                                05784
      rstlit();                                                        05785
      lplitreset ← FALSE;                                              05787
      END;                                                             05786
   IF NOT litreset THEN rstlit();                                      05525
   RETURN;                                                             05526
   END;                                                                05527
%..INITIALIZATION....%                                                 05528
   %..if csp is not valid, set to origin..%                            05529
```

```
   IF NOT da.daempty AND                                          05530
   NOT goodrng(da.dacsp) THEN                                     05531
      BEGIN                                                       05532
      da.dacsp.stpsid ← orgstid.stpsid;                           05533
      da.daccnt ← 1;                                              05534
      dafrmt(&da, 0);                                             05535
      RETURN;                                                     05536
      END;                                                        05537
%..terminal device check..%                                       05538
   IF nlmode = typewriter THEN err($"display attempted on
   typewriter terminal
   - proceed at own risk");                                       05539
   IF nldevice = devlproc AND cdtype = dspjpf AND
   (da.dalneighbor OR da.darneighbor OR da.datneighbor OR
   da.dabneighbor) THEN                                           05540
      RETURN (dafrmt (&da, 0));                                   05541
%..update previous file in da, for jump routines%                 05542
   da.dapstf ← da.dacsp.stfile;                                   05543
%..initialize marker flag to false..%                             05544
   mkrflg ← FALSE;                                                05545
%initialize first commands block to empty%                        05546
   cmdinit($commands, &da);                                       05547
%make sure aux big enough%                                        05548
   IF aulsize < da.dalsz THEN                                     05549
      BEGIN                                                       05550
      IF NOT (newaulsrt ← getblk(da.dalsz * lsrtl,
      $dspblk)) THEN %we will format into a table smaller
      than the old one - not intention of design%                 05551
         dismes(1,$"Possibly fatal DISPLAY ERROR...
         Enter a viewspec 'f' and check your screen.")            05552
      ELSE                                                        05553
         BEGIN                                                    05554
         freeblk(aulsrt - bhl, $dspblk); %deallocate old
         aux table%                                               05555
         aulsrt ← newaulsrt + bhl;%update global for aux
         LSRT%                                                    05556
         END;                                                     05557
      END;                                                        05558
%fix up lit area%                                                 05559
   IF nldevice = devlproc AND da.dalssup THEN                     05560
      BEGIN                                                       05561
      litup ← TRUE;                                               05562
      nextsup ← da.dalsrt + (da.dalssup := 0)*lsrtl;              05563
      END                                                         05564
   ELSE litup ← FALSE;                                            05565
   IF NOT litreset THEN rstlit();                                 05566
   lplitreset ← FALSE;                                            05788
%do empty screen and return%                                      05567
   IF da.daempty OR da.dacsp = endfil THEN                        05568
      BEGIN                                                       05569
      damt(&da);                                                  05570
      RETURN;                                                     05571
      END;                                                        05572
copyflag ← FALSE;                                                 05573
%..initialize LSRT (line segment reference table) pointers
to use auxiliary LSRT..%                                          05574
```

```
      oldlsrt ← rtsave ← tabltop ← newtop ← &art  ← da.dalsrt;
                                                                  05575
      arte ← da.dalsz * lsrtl + &art; %end + 1%                   05576
      rttop ← &rt ← aulsrt;                                       05577
      rte ← (rtsize ← aulsize) * lsrtl + aulsrt; %end + 1%
                                                                  05578
   %initialize LP links to 0 /                                     05773
      FOR &ls ← &art UP lsrtl UNTIL >= arte DO                    05771
         ls.rtlplink ← 0;                                         05772
   %..initialize vertical beam position..%                        05579
      da.dacrow ← 0;                                              05580
   %..clear auxillary lsrt..%                                     05581
      DO rt ← 0 UNTIL (&rt ← &rt + 1) = rte;                      05582
      &rt ← aulsrt;                                               05583
%..image update code..%                                           05584
   %..take care of frozen statements first...%                    05585
   IF da.davspec.vsfrzf THEN                                      05586
      BEGIN                                                       05587
      frzfrmt(&da);                                               05588
      %set effective top of LSRT past frozen statements%          05589
      DO IF art.rtsrce = srcdot THEN tabltop ← &art + lsrtl
                                                                  05590
      UNTIL NOT art.rtexis OR (&art ← &art + lsrtl) >= arte;
                                                                  05591
      END;                                                        05592
   %..start sequence at top of screen ..%                         05593
      &sa ← openseq(svstd ← da.dacsp, endstd ← seqend( svstd,
      sv1 ← da.davspec, sv2 ← da.davspc2 ), sv1, sv2,
      da.dausqcod, da.dacacode);                                  05594
      match ← FALSE; %no match between sequence & LSRT yet%
                                                                  05595
   %...MAIN FORMATTING LOOP....%                                  05596
      UNTIL da.dacrow > da.damrow OR (stid ← seqgen (&sa)) =
      endfil OR &rt >= rte                                        05597
      DO                                                          05598
         BEGIN                                                    05599
         IF stid = cdstd1 OR stid = cdstd2 THEN %must be
         reformatted%                                             05600
            BEGIN                                                 05601
            &ls ← &origls ← &rt;                                  05602
            stfrmt (&da, stid, 1, sa.swclvl, sa.swslvl,
            sa.swsvw, sa.swvspec, sa.swvsp2);                     05603
            IF srch (stid, sa.swclvl, match, tabltop) THEN
                                                                  05604
               BEGIN % This statement was on the screen
               before, see if we can eliminate some of the
               line segments%                                     05605
               &ols ← &art;                                       05606
               FOR &ls UP lsrtl UNTIL >= &rt DO                   05607
                  BEGIN  %compare corresponding line
                  segements, if they are the same and at same
                  location, dont rewrite them%                    05608
                  bp ← chbptr(ols.rtccnt - 1) + ( IF stid =
                  cdstd1 THEN $cdstr1 ELSE $cdstr2 );             05609
                  IF ls.rtxl = ols.rtxl
                  AND ls.rty = ols.rty
```

```
                AND ls.rtx2 = ols.rtx2
                AND (length ← slngth(ls.rtbps, ls.rtbpe)) =
                slngth(ols.rtbps, ols.rtbpe)
                AND complsg(ls.rtbps, bp, length) THEN %dont
                really need to write it!!%                      05610
                   BEGIN                                        05611
                   ls.rtlsid ← ols.rtlsid;                      05612
                   ls.rtnew ← FALSE;                            05613
                   copyflag ← TRUE;                             05614
                   END                                          05615
                ELSE %must really write this one, so make
                sure its neighbors on the same line also get
                written for LP's%                               05616
                   IF nldevice = devlproc THEN                  05617
                      BEGIN                                     05618
                      FOR &nls ← &ls - lsrtl DOWN lsrtl
                      UNTIL < &origls DO                        05619
                         IF nls.rty = ls.rty THEN               05620
                            BEGIN                               05621
                            IF NOT nls.rtnew THEN               05622
                               BEGIN                            05623
                               nls.rtnew ← TRUE;                05624
                               nls.rtlsid ← 0;                  05625
                               END;                             05626
                            END                                 05627
                         ELSE EXIT LOOP;                        05628
                      FOR &nls ← &ls + lsrtl UP lsrtl UNTIL
                      >= &rt DO                                 05629
                         IF nls.rty = ls.rty THEN               05630
                            BEGIN                               05631
                            &ols ← &ols + lsrtl;                05632
                            &ls ← &nls;                         05633
                            IF &ols >= arte THEN EXIT LOOP;
                                                                05634
                            END                                 05635
                         ELSE EXIT LOOP;                        05636
                      END;                                      05637
                   &ols ← &ols + lsrtl;                         05638
                   IF &ols >= arte THEN EXIT LOOP;              05639
                   END;                                         05640
             END;                                               05641
          END                                                   05642
       ELSE                                                     05643
          BEGIN %search for match in old LSRT%                  05644
          IF (match ← srch (stid, sa.swclvl, match, tabltop
          : newtop)) THEN                                       05645
             BEGIN                                              05646
             IF newtop AND stid NOT= cdstop THEN tabltop ←
             newtop;                                            05647
             IF (litup AND &art < nextsup) OR NOT
             (lscopy(TRUE,&da)) THEN                            05648
                stfrmt (&da, stid, 1, sa.swclvl, sa.swslvl,
                sa.swsvw, sa.swvspec, sa.swvsp2)                05649
                   %reformat if partial stmt found or
                   re"toring supressed statements%              05650
             ELSE copyflag ← TRUE;                              05651
```

```
      END                                                    05652
   ELSE %no match%                                           05653
      stfrmt (&da, stid, 1, sa.swc1v1, sa.sws1v1,
      sa.swsvw, sa.swvspec, sa.swvsp2);                      05654
   END;                                                      05655
IF stid = cdstop AND cdstop # endfil THEN                    05656
   %no need to reformat (stfrmt) rest of LSRT%               05657
   BEGIN                                                     05658
   %if old table has useful information, copy all of
   it%                                                       05659
      IF srch(stid, sa.swc1v1, match, tab1top) THEN
                                                             05660
         BEGIN                                               05661
         &art ← 1stidnxt (&art);                             05662
            %bump lsrt source addr%                          05663
         match ← TRUE;                                       05664
         copyflag ← TRUE;                                    05665
         lscopy(FALSE,&da); %copy rest of LSRT%              05666
         END                                                 05667
      ELSE match ← FALSE; %flag for no copy done%
                                                             05668
   IF da.dacrow <= da.damrow AND &rt < rte THEN %more
   room on screen%                                           05669
      BEGIN %get and reformat new statements%                05670
      IF match THEN %copy caused sequence to be
      broken%                                                05671
         BEGIN                                               05672
         % save viewspecs from sequence work area in
         case user seq code canged them. %                   05673
            sv1 ← sa.swvspec;                                05674
            sv2 ← sa.swvsp2;                                 05675
         %save base level from sequence work area in
         case changed by impending openseq%                  05676
            s1v1 ← sa.sws1v1;                                05677
         %close last sequence%                               05678
            closeseq(&sa);                                   05679
         %open new sequence at last stid copied%             05680
            &sa ← openseq (/&rt - 1srt1/.rtstid,
            endstd,  sv1, sv2, da.dausqcod,
            da.dacacode);                                    05681
            stid ← seqgen(&sa);%already
            formatted-throw away%                            05682
         %restore base level for this screen%                05683
            sa.sws1v1 ← s1v1;                                05684
         END;                                                05685
      %do regular format on stmts now able to fit on
      screen for first time%                                 05686
         UNTIL &rt >= rte OR da.dacrow > da.damrow OR
         (stid ← seqgen (&sa)) = endfil DO                   05687
            stfrmt (&da, stid, 1, sa.swc1v1,
            sa.sws1v1, sa.swsvw, sa.swvspec,
            sa.swvsp2);                                      05688
      END;                                                   05689
   EXIT;                                                     05690
   END; %case where old screen can be used%                  05691
END; %..OF MAIN LOOP....%
```

```
                                                                    05692
%close whatever sequence it is on%                                  05693
   da.davspec ← sa.swvspec;                                         05694
   da.davspc2 ← sa.swvsp2;                                          05695
   dspvsp(da.davspec, da.davspc2, 3);                               05696
   closeseq(&sa);                                                   05697
%construct movelsg and deletlsg commands and update LSRT%
                                                                    05698
   IF copyflag THEN                                                 05699
      BEGIN                                                         05700
      %initialize pointers%                                         05701
         &art ← rttop;                                              05702
         arte ← rtsize * lsrtl + rttop;                             05703
         rte ← da.dalsz  * lsrtl + rtsave;                          05704
      UNTIL NOT art.rtexis OR                                       05705
      &art >= arte DO                                               05706
         BEGIN                                                      05707
         IF NOT art.rtnew THEN                                      05708
            BEGIN                                                   05709
            &rt ← rtsave; %old table%                               05710
            DO                                                      05711
               IF art.rtlsid = rt.rtlsid AND                        05712
                  art.rty NOT= rt.rty THEN                          05713
                  BEGIN                                             05714
                  pushdc(movelsg, &art);                            05715
                  IF nldevice NOT= devlproc THEN                    05777
                     cklsfreestring(&art);                          05716
                        %moving causes the first byteptr to
                        be set -1, so we must free th
                        string now (while we still know
                        where it is)%                               05717
                  EXIT;                                             05718
                  END                                               05719
            UNTIL (&rt ← &rt + lsrtl) >= rte OR                     05720
            NOT rt.rtexis;                                          05721
            END;                                                    05722
         &art ← &art + lsrtl;                                       05723
         END;                                                       05724
      %construct deletlsg commands%                                 05725
         &rt ← rtsave; %old table%                                  05726
         DO                                                         05727
            BEGIN                                                   05728
            &art ← rttop;                                           05729
            deleted ← TRUE;                                         05730
            DO                                                      05731
               IF rt.rtlsid = art.rtlsid THEN                       05732
                  BEGIN                                             05733
                  deleted ← FALSE;                                  05734
                  EXIT;                                             05735
                  END                                               05736
            UNTIL (&art ← &art + lsrtl) >= arte OR
            NOT art.rtexis;                                         05737
            IF deleted THEN                                         05738
               BEGIN                                                05739
               cklsfreestring(&rt); %see if it is allocated
               string to deallocate%                                05740
```

```
                        pushdc(deletlsg, &rt);                          05741
                        END;                                            05742
                     END                                                05743
                  UNTIL (&rt ← &rt + lsrtl) >= rte OR
                  NOT rt.rtexis;                                        05744
               END                                                      05745
            ELSE %no lsrt entries were copied, thus just clear out
            old image and free any strings%                             05746
               BEGIN                                                    05747
               pushdc(blankda, 0);                                      05748
               &art ← rtsave; %arte is still set up%                    05749
               DO cklsfreestring(&art)                                  05750
                  UNTIL (&art ← &art + lsrtl) >= arte OR NOT
                  art.rtexis;                                           05751
               END;                                                     05752
            %process any commands that were stacked%                    05753
               prcmds(&da);                                             05754
            %..clear rest of current LSRT...%                           05755
               BUMP DOWN &rt;                                           05756
               UNTIL (&rt ← &rt + 1) >= rte DO rt ← 0;                  05757
      %..termination..%                                                 05758
         IF da.dacrow = 0  OR                                           05759
         (da.dacrow = da.davinc AND                                     05760
         rttop.rtsrce = srcdot) THEN                                    05761
            damt(&da);                                                  05762
      %transpose table pointers so aux becomes real LSRT%               05763
         aulsrt ← rtsave; %old da.dalsrt%                               05764
         da.dalsrt ← rttop; %current (new) table%                       05765
         aulsize ← da.dalsz; %old size%                                 05766
         da.dalsz ← rtsize; %new size%                                  05767
      RETURN;                                                           05768
      END.                                                              05769
                                                                        05770

   (cklsfreestring)PROCEDURE(ls); %if line seg is from string
   allocated from dspblk then deallocate it%                             0378
      REF ls;                                                            0379
      CASE ls.rtsrce OF                                                  0380
         =srcstno, =srcsig:                                              0381
            freestring (ls.rtbps.bpadr -1, $dspblk); %freestring
            checks bounds for calls involving dspblk%                    0382
         ENDCASE NULL;                                                   0383
      RETURN; END.                                                       0384
                                                                         0385
   (damt)      PROCEDURE (da); %empty da routine%                        0386
      %Thise routine constructs the empty message in a text              0387
      display area%                                                      0388
      %-----------%                                                      0389
      LOCAL ls, end;                                                     0390
      REF ls, da;                                                        0391
      IF da.dacrow > 0 THEN                                              0392
         BEGIN                                                           0393
         clearda(&da); %clear display area %                             0394
         clrall(&da, TRUE);%zap lsrt entries%                            0395
         END;                                                            0396
      &rt ← da.dalsrt;                                                   0397
      end ← &rt + lsrtl;                                                 0398
```

```
    DO rt ← 0 UNTIL (&rt ← &rt + 1) = end;                          0399
    &rt ← &rt - lsrtl;                                              0400
    rt.rtexis ← rt.rtnew ← rt.rtllsg ← TRUE;                        0401
    rt.rtsrce ← srcast;                                             0402
    rt.rtfont ← fnormal;                                            0403
    rt.rtstid.stastr ← TRUE;                                        0404
    rt.rtstid.stadr ← $mt;                                          0405
    rt.rtcent ← 1;                                                  0406
    rt.rthinc ← da.dahinc;                                          0407
    rt.rtcsize ← da.dacsize;                                        0408
    rt.rtx1 ← da.dahinc;                                            0409
    rt.rtx2 ← gapcol ←  MAX(rt.rtx1, MIN(rt.rtx1 +
    (mt.L-1)*da.dahinc, da.daright-da.daleft));                     0410
    rt.rty ← da.dacrow ← 0;                                         0411
    rt.rtbps ← chbmty + $mt;                                        0412
    rt.rtbpe ← gapbp ← chbptr(mt.L) + $mt;                          0413
    wrtstr(&da, &rt);                                               0414
    RETURN;                                                         0415
    END.                                                            0416
                                                                    0417
(frzfrmt)  PROCEDURE (da);  %display frozen statements%             0418
    %This routine formats the frozen statements for a               0419
    display area%                                                   0420
    %-------------------%                                          0421
    LOCAL  ls, fz, davs1, davs2, sa, stid;                          0422
    REF  ls, da, fz, sa;                                            0423
    %display the frozen statements if viewspec is                   0424
    on--otherwise return%                                           0425
       IF NOT da.davspec.vsfrzf THEN RETURN;                        0426
    davs1 ← da.davspec; %save view specs from display area record%
                                                                    0427
    davs2 ← da.davspc2;                                             0428
    IF &fz ← da.dafrz1 THEN %there are frozen statements%           0429
       DO                                                           0430
          IF fz.fzexis THEN                                         0431
             BEGIN  %statement frozen...display it%                 0432
             % set up a sequence that will only return the first
             statement if it passes -- by setting the L viewspec
             to E + 0 %                                             0433
                da.davspec ← <PRMSPC, setlt>('e, fz.fzvspec,
                fz.fzvspc2 : da.davspc2);                           0434
             &sa ← openseq (fz.fzstid, fz.fzstid, da.davspec,
             da.davspc2, da.dausqcod, da.dacacode);                 0435
             UNTIL (stid ← seqgen (&sa)) = endfil DO                0436
                stfrmt(&da, stid, 1, sa.swclvl, sa.swslvl,
                sa.swsvw, sa.swvspec, sa.swvsp2);                   0437
             closeseq (&sa);                                        0438
             IF da.dacrow > da.damrow THEN RETURN; %filled display
             area%                                                  0439
             END                                                    0440
          ELSE err($"NLS error: Frozen statement list incorrect
          Turn off viewspec e")                                     0441
       UNTIL (&fz ← fz.fznext) = 0;                                 0442
    da.davspec ← davs1; %restore original view specs%               0443
    da.davspc2 ← davs2;                                             0444
    %display dotted line%                                           0445
```

```
        rt.rtbps ← chbmty + $dotstr;                                    0446
        rt.rtbpe ← gapbp ← chbptr(dotstr.L) + $dotstr;                  0447
        rt.rty ← da.dacrow := da.dacrow + da.davinc;                    0448
        rt.rtx1 ← 0;                                                    0449
        rt.rtx2 ← gapcol ←  MAX(rt.rtx1, MIN(rt.rtx1 +
        (dotstr.L-1)*da.dahinc, da.daright-da.daleft));                 0450
        rt.rtlisg ← rt.rtexis ← TRUE;                                   0451
        rt.rtcsize ← da.dacsize;                                        0452
        rt.rthinc ← da.dahinc;                                          0453
        rt.rtsrce ← srcdot;                                             0454
        pushdc(writelsg, &rt);                                          0455
        nxtlsg(&da);                                                    0456
        RETURN;                                                         0457
        END.                                                            0458
                                                                        0459
%....Display format support....%                                        0460
    (stfrmt)   PROC (da, stid, charct, level, origlev,  stvec,
    viewspec1, vwspc2);  %statement format%                             05798
        %This routine formats a statement according to the information
        in the text display area record, da, the level of the
        statement, level, and the statement vector work area, stvec.
        The formating begins at the character indicated by charct.%
                                                                        05799
        %NOTE:  special handling of indentation--if indent=OFF and
        branch/plex only, integrity of structure maintained%           05800
        %viewspec1 is first word of viewspecs, as obtained from
        seqgen%                                                         05801
        %The global 'rt' is the current line segment referencetable
        entry%                                                          05802
        %--------------%
                                                                        05803
        LOCAL                                                           05804
            bptr,              %temp for saving byte pointer values%
            stringaddr,         %temp for string address values%        05805
            numberadr,         %temp for stmt # address%                05806
            char,              %return from lsgfrmt%                    05807
            numflg,                                                     05808
            sdbadr,                                                     05809
            word,                                                       05810
            trunc,               %current line for truncation viewspecxx%
                                                                        05811
            tabcol,                                                     05812
            indent,              %left margin offset%                   05813
            ls,                  %line segment address%                 05814
            header,                                                     05815
            length,                                                     05816
            right,                                                      05817
            left,                                                       05818
            blnklnf,    %flag to note blank line already out%           05819
            wa[10];                                                     05820
        REF da, sdbadr, ls, stringaddr, numberadr;
                                                                        05821
        ON SIGNAL                                                       05822
            =spacerr: %could not allocate such a large block%           05823
                BEGIN                                                   05824
                rt.rtexis ← FALSE; %make last one not exist in case it
```

```
        is incomplete%                                                  05825
        GOTO stfrend; %entry at &rt has not been pushed on
        commands list yet%                                              05826
        END;                                                            05827
ELSE NULL;

                                                                        05828
IF NOT da.daexis THEN                                                   05829
   err($"Fatal display error in STFRMT");                               05830
IF stid = endfil THEN RETURN;                                           05831
dgstid ← stid;                                                          05832
IF NOT stid.stastr THEN                                                 05833
   BEGIN                                                                05834
   lodsdb(getsdb(stid) : &sdbadr);                                      05835
   dgstml ← sdbadr.schars;                                              05836
   END                                                                  05837
ELSE dgstml ← /stid.stadr/.L;                                           05838
lastch ← ENDCHR;                                                        05839
   %for special case of null line%                                      05840
indent ←        %indentation%                                           05841
   CASE TRUE OF                                                         05842
      = viewspec1.vsindf:                                               05843
         MAX (0, MIN (da.daind * (level-1), da.damind));                05844
      = da.davspec.vsbrof, = da.davspec.vsplxf:                         05845
         MAX (0, MIN (da.daind * (level-origlev), da.damind));
                                                                        05846
      ENDCASE 0;                                                        05847
IF charct <= 1 THEN                                                     05848
   BEGIN                                                                05849
   charct ← 1;                                                          05850
   IF NOT viewspec1.vsnamf THEN %do not show statement name%
                                                                        05851
      IF stid.stastr THEN %a-string%                                    05852
         BEGIN                                                          05853
         IF NOT da.dacsp.stastr THEN                                    05854
            BEGIN %try to skip over text that looks like a
            name%                                                       05855
            %this is bullshit code--probably should be removed
            or replaced%                                                05856
            wa ← stid;                                                  05857
            wa/1/ ← 1;                                                  05858
            fechc1(forward, $wa);                                       05859
            header ← filhdr(da.dacsp.stfile);                           05860
               %wont work for frozen statement a-strings                05861
               from another file%                                       05862
            left ← /$namd11-$filhed+header/;                            05863
            right ← /$namd12-$filhed+header/;                           05864
            xtrnam($stn, $wa, left, right);                             05865
            IF stn.L = empty THEN charct ← 1 %no name found%
                                                                        05866
            ELSE charct ← wa/1/;                                        05867
            END;                                                        05868
         END                                                            05869
      ELSE charct ← sdbadr.sname;%skip over name%                       05870
   END;                                                                 05871
dgstl ← charct - 1;                                                     05872
%initialize LSRT entry%                                                 05873
```

```
      rt.rtxl ← 0;                                                    05874
      rt.rty ← da.dacrow;                                             05875
      rt.rtexis ← rt.rtnew ← TRUE;                                    05876
      rt.rtlisg ← FALSE;                                              05877
      rt.rtlsid ← 0;                                                  05878
      rt.rthinc ← da.dahinc;                                          05879
      rt.rtcsize ← da.dacsize;                                        05880
      rt.rtstid ← stid;                                               05881
      rt.rtccnt ← charct;                                             05882
      rt.rtcbug ← 1;                                                  05883
      rt.rtsrce ← srcnul;                                             05884
      rt.rtfont ← fnormal;                                            05885
      rt.rtlev ← level;                                               05886
      rt.rtx2 ← da.damcol / rt.rthinc * rt.rthinc; %round down
      damcol for stop coord%                                          05887
      rt.rtpartst ← TRUE;%assume this segment not last in
      statement%                                                      05888
      %set up byte pointers for reading characters from string or
      statement%                                                      05889
      IF NOT stid.stastr THEN                                         05890
         rt.rtbps ← rt.rtbpe ← stbptr(charct - 1) + &sdbadr +
         sdbhdl                                                       05891
      ELSE                                                            05892
         rt.rtbps ← rt.rtbpe ← chbptr(charct - 1) + stid.stadr;
                                                                      05893
   IF NOT stid.stastr AND viewspecl.vsstnf AND
   (stid.stpsid NOT= origin OR viewspecl.vssidf) THEN                 05894
      BEGIN                                                           05895
      &numberadr ← getstring(30, $dspblk);                            05896
      IF viewspecl.vssidf  % display line #'s as sids %               05897
         THEN                                                         05898
            *numberadr* ← '0, STRING( getsid(stid) )                  05899
         ELSE                                                         05900
            fechnm(stvec, &numberadr);                                05901
               %convert statement vector to string%                   05902
      IF NOT viewspecl.vsstnr THEN                                    05903
         BEGIN %statement number on left%                             05904
         numflg ← FALSE;                                              05905
         rt.rtbps ← chbmty + &numberadr;                              05906
         rt.rtbpe ← chbptr(numberadr.L) + &numberadr;                 05907
         rt.rtsrce ← srcstno;                                         05908
         rt.rthinc ← da.danohi;                                       05909
         rt.rtcsize ← da.danccs;                                      05910
         gapcc ← 0;                                                   05911
         gapcol ← da.daccol;                                          05912
         gapbp ← rt.rtbpe;                                            05913
         char ←  dgstl; %save current char count%                     05914
         &ls ← nxtlsg(&da);                                           05915
         dgstl ←  char; %restore current char count%                  05916
         ls.rtxl ← indent;                                            05917
         ls.rtx2 ← MIN(ls.rtxl + (numberadr.L-1) * ls.rthinc,
         da.damcol - ls.rthinc); %for bug selection%                  05918
         IF da.damcol - ls.rtxl < numberadr.L*ls.rthinc THEN          05919
            (setxcm);                                                 05920
            *numberadr*/(da.damcol - ls.rtxl)/ls.rthinc/ ← '!;
                                                                      05921
```

```
            %to note that some of stmt # is lost%                    05922
         ls.rtllsg ← FALSE;                                          05923
         rt.rtllsg ← FALSE;                                          05924
         IF ls.rtx2 + 2*da.dahinc < da.damcol THEN                   05925
            da.daccol ← rt.rtxl ←
                  ls.rtx2 + 2*da.dahinc - indent                     05926
         ELSE                                                        05927
            BEGIN                                                    05928
            ls.rtllsg ← TRUE;                                        05929
            rt.rty ← da.dacrow := da.dacrow + da.davinc;             05930
            rt.rtxl ← da.daccol ← 0;                                 05931
            END;                                                     05932
         pushdc(writelsg, &ls);                                      05933
         rt.rtcsize ← da.dacsize;                                    05934
         rt.rthinc ← da.dahinc;                                      05935
         rt.rtbps ← rt.rtbpe ← stbptr(charct - 1) + &sdbadr +
         sdbhdl;                                                     05936
         END                                                         05937
      ELSE %numbers on right%                                        05938
         numflg ← TRUE;                                              05939
      END                                                            05940
   ELSE numflg ← FALSE; %no statement number%                        05941
   blnklnf ← FALSE; %init; TRUE if stmt # at right on separate
   line%                                                             05942
   IF viewspecl.vsmkrf AND NOT stid.stastr THEN                      05943
      BEGIN %snow markers%                                           05944
      &mkrptr ← $mkrtb - $filhed +                                   05945
         (header ← filhdr(stid.stfile));                             05946
         % initialize to beginning of marker table %                 05947
      mkrend ← mkrptr + ($mkrtbl-$filhed+header)*mkrl;               05948
         % current end of table %                                    05949
      % initialize flag to false--do not look for markers           05950
      in this statement %                                            05951
         mkrflg ← FALSE;                                             05952
      DO %look for this statement's psid in the marker             05953
      table%                                                         05954
         IF mkrptr.mkpsid = stid.stpsid THEN                         05955
            BEGIN   % found one %                                    05956
            % set 'look-for-markers' flag for this                  05957
            statement%                                               05958
               mkrflg ← TRUE;                                        05959
            %save chr count + 1%                                     05960
               mkrcnt ← mkrptr.mkccnt + 1;                           05961
            % terminate loop %                                      05962
               EXIT;                                                 05963
            END                                                      05964
      UNTIL (mkrptr ← mkrptr + mkrl) > mkrend;                       05965
      END                                                            05966
   ELSE mkrflg ← FALSE;                                              05967
   %initialize truncation counter%                                   05968
      trunc ← viewspecl.vstrnc;                                      05969
   UNTIL (trunc := trunc - 1) <= 0 OR
   &rt >= rte OR
   da.dacrow > da.damrow DO %format lines for the statement%         05970
      (stfrnegin):                                                   05971
      BEGIN %once per line%                                          05972
```

```
rt.rtxl ← da.daccol ← indent + rt.rtxl;                          05973
rt.rty ← da.dacrow := da.dacrow + da.davinc;                     05974
%format a line%                                                  05975
   UNTIL &rt >= rte DO                                           05976
      BEGIN                                                      05977
      rt.rtsrce ← srcstat;                                       05978
      CASE (char ← lsgfrmt(&da)) OF                              05979
         = ENDCHR: %end of input%                                05980
            BEGIN  %finish up and return%                        05981
            IF (NOT numflg OR (numflg AND da.daccol >=
            rt.rtx2 - (numberadr.L)*rt.rthinc)) THEN
            rt.rtllsg ← TRUE;                                    05982
               %if statement number to be processed, this
               is not last segment in this line, otherwise
               it is%                                            05983
            rt.rtexis ← TRUE;                                    05984
            &ls ← nxtlsg(&da);                                   05985
            IF rt.rtllsg AND NOT (numflg OR
            viewspecl.vsidtf OR viewspecl.vsblkf) THEN
            ls.rtpartst ← FALSE;                                 05986
            %statement complete %                                05987
            pushdc(writelsg, &ls);                               05988
            trunc ← 0;                                           05989
            EXIT;                                                05990
            END;                                                 05991
         = SP, = CR: %end of line%                               05992
            BEGIN  %finish up and do next line%                  05993
            rt.rtexis ← TRUE;                                    05994
            IF trunc > 0 OR (NOT numflg OR (numflg AND
            da.daccol >= rt.rtx2 -
            (numberadr.L)*rt.rthinc)) THEN rt.rtllsg ←
            TRUE;                                                05995
            &ls ← nxtlsg(&da);                                   05996
            IF (trunc <= 0 AND rt.rtllsg) AND NOT (numflg
            OR viewspecl.vsidtf OR viewspecl.vsblkf) THEN
            ls.rtpartst ← FALSE; %statement now complete%
                                                                 05997
            pushdc(writelsg, &ps);                               05998
            rt.rtxl ← 0;                                         05999
            EXIT;                                                06000
            END;                                                 06001
         = TAB: %a TAB was input%                                06002
            BEGIN  %continue if not end of line%                 06003
            %save global values across nxtlsg call -
            kludge%                                              06004
            char ← gapcc;                                        06005
            bptr ← gapcol;                                       06006
            %stfrmt starts daccol off at 0, while fndtab
            uses da.dahinc for first position.  param to
            fndtab and return from fndtab have to be
            converted by one horizontal inc%                     06007
            tabcol ← fndtab(&da, da.daccol/rt.rthinc) *
            rt.rthinc;                                           06008
            IF tabcol > rt.rtx2 THEN                             06009
               BEGIN %too many columns--start new                06010
               line%                                             06011
```

```
            cdbckc();                                          06012
            da.daccol ← da.daccol - rt.rthinc;
            %apparently to satisfy lsgfrmt%                    06013
            rt.rtllsg ← TRUE; %last segment, this line%
                                                               06014
            &ls ← nxtlsg(&da);                                 06015
            gapcc ← char;                                      06016
            gapcol ← bptr;                                     06017
            ls.rtexis ← TRUE;                                  06018
            pushdc(writelsg, &ls);                             06019
            rt.rtx1 ← 0; %start next tab at left +
            indent%                                            06020
            EXIT;                                              06021
            END;                                               06022
         &ls ← nxtlsg(&da);                                    06023
         gapcc ← char;                                         06024
         gapcol ← bptr;                                        06025
         ls.rtexis ← TRUE;                                     06026
         pushdc(writelsg, &ls);                                06027
         da.daccol ← rt.rtx1 ← tabcol - rt.rthinc;
         %convert fndtab return%                               06028
         END;                                                  06029
      ENDCASE                                                  06030
         BEGIN %make separate lsg with non-printing
         string %                                              06031
         %check if adding non-printing char string will
         exceed current line%                                  06032
         bptr ← gapbp; %save old rtbpe value for lsg
         after non-printing char%                              06033
         IF 4 * da.dahinc + da.daccol > rt.rtx2 THEN
                                                               06034
            BEGIN                                              06035
            cdbckc(); %backup input text pointer%              06036
            rt.rtexis ← TRUE;                                  06037
            IF trunc > 0 OR (NOT numflg OR (numflg AND
            da.dacccl >= rt.rtx2 -
            (numberadr.L)*rt.rthinc)) THEN rt.rtllsg ←
            TRUE;                                              06038
            &ls ← nxtlsg(&da);                                 06039
            IF (trunc <= 0 AND rt.rtllsg) AND NOT
            (numflg OR viewspecl.vsidtf OR
            viewspecl.vsblkf) THEN ls.rtpartst ← FALSE;
            %statement now complete%                           06040
            pushdc(writelsg, &ls);                             06041
            rt.rtx1 ← da.daccol ← indent;                      06042
            rt.rty ← da.dacrow := da.dacrow + da.davinc;
                                                               06043
            END                                                06044
         ELSE                                                  06045
            BEGIN                                              06046
            &ls ← nxtlsg(&da); %get next line segment%
                                                               06047
            ls.rtexis ← TRUE;                                  06048
            pushdc (writelsg, &ls); %write previous lsg%
                                                               06049
            END;                                               06050
```

```
                    %set pointers to appropriate non-printing char
                    string%                                          06051
                    rt.rtbps ← chbmty + (&stringaddr ←
                    npstrad(char));                                  06052
                    rt.rtexis ← TRUE; %mark the entry as existing%
                                                                     06053
                    gapbp ← chbptr (stringaddr.L) + &stringaddr;
                                                                     06054
                    rt.rtcbug ← stringaddr.L; %make whole fat
                    character buggable%                              06055
                    da.daccol ←  stringaddr.L * rt.rthinc +
                    rt.rtxl;                                         06056
                    gapcol ← da.daccol - rt.rthinc;                  06057
                    gapcc ← dgstl + 1; %advance past non-printing
                    char%                                            06058
                    &ls ← nxtlsg(&da);                               06059
                    rt.rtbps ← rt.rtbpe ← bptr; %restore old rtbpe
                    value pointing to user file%                     06060
                    IF trunc <= 0 AND NOT(viewspecl.vsblkf OR
                    viewspecl.vsidtf OR numflg) THEN ls.rtpartst ←
                    FALSE;                                           06061
                    pushdc (writelsg, &ls); %write previous lsg%
                                                                     06062
                    END;                                             06063
          END;                                                       06064
    END;                                                             06065
%clear excess line segment%                                          06066
    (strend):                                                        06067
    rt.rtbpe ← rt.rtbps;                                             06068
%***IF NOT cdvect() THEN RETURN(FALSE);***%                          06069
IF numflg AND trunc <= 0 AND NOT (da.daccol >= da.damcol -
rt.rthinc*(numberadr.L+1) AND da.dacrow > da.damrow) THEN             06070
    BEGIN %statement # on right at end of stmt unless both line
    and screen full%                                                 06071
    numflg ← FALSE;                                                  06072
    bptr ← rt.rtbps;                                                 06073
    rt.rtbps ← chbmty + &numberadr;                                  06074
    rt.rtbpe ← chbptr(numberadr.L) + &numberadr;                     06075
    rt.rtsrce ← srcstno;                                             06076
    rt.rthinc ← da.danohi;                                           06077
    rt.rtcsize ← da.danocs;                                          06078
    char ← dgstl; %save old character count value used for
                  sensing end of input%                              06079
    rt.rtx2 ← da.damcol - da.dahinc;                                 06080
    rt.rtxl ← MAX(rt.rtx2 - (numberadr.L -1) * rt.rthinc, 0);
                                                                     06081
    IF da.damcol < numberadr.L*rt.rthinc THEN                        06082
       (setxcl):                                                     06083
       *numberadr*[da.damccl /rt.rthinc] ← '!;                       06084
    IF da.daccol >= rt.rtxl THEN                                     06085
       BEGIN  %have to put number on separate line%                  06086
       rt.rty ← da.dacrow := da.dacrow + da.davinc;                  06087
       blnklnf ← TRUE;                                               06088
       END                                                           06089
    ELSE blnklnf ← FALSE;                                            06090
    IF NOT viewspecl.vsidtf THEN                                     06091
```

```
      BEGIN   %statement complete%                                    06092
      rt.rtpartst ← FALSE;                                            06093
      rt.rtllsg ← TRUE;                                               06094
      END                                                             06095
   ELSE IF da.daccol < rt.rtx1 THEN rt.rtllsg ← TRUE;                 06096
   gapcc ← 0;                                                         06097
   gapcol ← da.daccol ← da.damcol;                                    06098
   gapbp ← rt.rtbpe;                                                  06099
   &ls ← nxtlsg(&da);                                                 06100
   ls.rtexis ← TRUE;                                                  06101
   rt.rtx2 ← da.damcol - da.dahinc;                                   06102
   rt.rtx1 ← 0;                                                       06103
   rt.rtllsg ← FALSE;                                                 06104
   rt.rtcsize ← da.dacsize;                                           06105
   rt.rthinc ← da.dahinc;                                             06106
   rt.rtbps ← rt.rtbpe ← bptr;                                        06107
   IF NOT (blnklnf AND viewspecl.vsidtf AND da.dacrow <=
   da.damrow) THEN pushdc(writelsg, &ls);     %May not really
   be able to put stmt # and signatures on same extra line;
   delay putting this line segment out until you find out
   unless the display is full%                                        06108
   rt.rtccnt ← (dgstl ← char) + 1; %restore old dgstl value -
   current char count%                                                06109
   END;                                                               06110
IF viewspecl.vsidtf AND NOT stid.stastr  THEN                         06111
   BEGIN                                                              06112
   IF da.dacrow <= da.damrow  THEN                                    06113
      BEGIN                                                           06114
      &stringaddr ← getstring(30, $dspblk);                           06115
      rt.rtexis ← TRUE;                                               06116
      rt.rtllsg ← TRUE;                                               06117
      IF NOT blnklnf THEN rt.rty ← da.dacrow := da.dacrow +
      da.davinc;                                                      06118
      *stringaddr* ← NULL;                                            06119
      fechsig(stid, &stringaddr);                                     06120
      rt.rtbps ← chbmty + &stringaddr;                                06121
      rt.rtbpe ← chbptr(stringaddr.L) + &stringaddr;                  06122
      rt.rtsrce ← srcsig;                                             06123
      rt.rthinc ← da.dasghi;                                          06124
      rt.rtcsize ← da.dasgcs;                                         06125
      IF blnklnf THEN                                                 06126
         BEGIN                                                        06127
         IF ls.rtx1 - (stringaddr.L+3)*ls.rthinc  < indent
         THEN                                                         06128
            BEGIN                                                     06129
            blnklnf ← FALSE;                                          06130
            ls.rtllsg ← TRUE;                                         06131
            rt.rty ← da.dacrow := da.dacrow + da.davinc;              06132
            END;                                                      06133
         pushdc(writelsg, &ls);                                       06134
         END;                                                         06135
      IF blnklnf THEN rt.rtx2 ← da.daccol ← ls.rtx1 -
      (3*da.dahinc)                                                   06136
      ELSE rt.rtx2 ← da.daccol ← da.damcol - da.dahinc;               06137
      rt.rtx1 ←                                                       06138
         MAX(indent, rt.rtx2 - (stringaddr.L*rt.rthinc));             06139
```

```
            gapcc ← 0;                                                  06140
            gapcol ← rt.rtx2;                                           06141
            gapbp ← rt.rtbpe;                                           06142
            rt.rtpartst ← FALSE; %statement complete%                   06143
            &ls ← nxtlsg(&da);                                          06144
            pushdc(writelsg, &ls);                                      06145
            END;                                                        06146
         RETURN;                                                        06147
         END                                                            06148
      ELSE IF viewspecl.vsblkf AND NOT blnkinf THEN                     06149
         BEGIN                                                          06150
         IF da.dacrow <= da.damrow THEN                                 06151
            BEGIN                                                       06152
            rt.rtexis ← TRUE;                                           06153
            rt.rtlsg ← TRUE;                                            06154
            rt.rty ← da.dacrow := da.dacrow + da.davinc;                06155
            rt.rtpartst ← FALSE; %statement complete%                   06156
            rt.rtsrce ← srcnul;                                         06157
            %clear line segment%                                        06158
               rt.rtbpe ← rt.rtbps ← chbmty;                            06159
               rt.rtx1 ← 0;                                             06160
            gapcol ← da.daright;                                        06288
            gapbp ←chbmty;                                              06289
            &ls ← nxtlsg(&da);                                          06161
            pushdc(writelsg, &ls);                                      06162
            END;                                                        06163
         RETURN;                                                        06164
         END;                                                           06165
      %mkrsho();%                                                       06166
      IF trunc <= 0 AND NOT viewspecl.vsidtf AND NOT
      viewspecl.vsblkf AND NOT numflg THEN ls.rtpartst ← FALSE;         06167
      IF stid.stastr THEN                                               06168
         BEGIN   %must process commands accumulated so far to avoid
         losing the contents of the string%                             06169
         prcmds(&da);                                                   06170
         cmdinit($commands, &da); %zap the command list%                06171
         END;                                                           06172
      RETURN;                                                           06173
      END.                                                              06174
   (lsgfrmt)' PROCEDURE (da);  %line segment formatter%                 06175
      %Reasons for terminating a line segment:                          06176
         end of input: RETURN(ENDCHR),                                  06177
         end of allowed columns: RETURN(SP),                            06178
         CR was read: RETURN(CR),                                       06179
         TAB was read: RETURN(TAB),                                     06180
         control character was read: RETURN(the character),             06181
         font change: RETURN(swchl)%                                    06182
      %assumes globals dgstl, dgstml%                                   06183
      %------------------%                                             06184
      LOCAL bptr, char, col, hinc, x2lim;                               06185
      REF da;                                                           06186
      rt.rtexis ← TRUE;                                                 06187
      %extract important variables%                                     06188
      hinc ← rt.rthinc;                                                 06189
      x2lim ← rt.rtx2/hinc*hinc;                                        06190
      bptr ← rt.rtbpe;                                                  06191
```

```
col ← da.daccol - hinc;                                          06192
UNTIL (col ← col + hinc) > x2lim DO                              06193
   BEGIN                                                         06194
   IF (dgstl ← dgstl + 1) > dgstml THEN                          06195
      BEGIN                                                      06196
      %input exhausted%                                          06197
      gapbp ← bptr;                                              06198
      gapcol ← col - hinc;                                       06199
      gapcc ← dgstl-1;                                           06200
      IF col = rt.rtx1 THEN                                      06201
         BEGIN %line segment is empty%                           06202
         %global lastch is used to determine if null line
         segment implies a null statement--a null statement's
         first char should be bugable%                           06203
         IF lastch NOT= ENDCHR THEN                              06204
            rt.rtsrce ← srcnul; %not a null statement--make it
            unbugable%                                           06205
         rt.rtx2 ← rt.rtx1; %only first char is bugable%         06206
         END;                                                    06207
      lastch ← char ← ENDCHR;                                    06208
      GOTO lsfout;                                               06209
      END;                                                       06210
   CASE char ← ↑bptr OF %get next character from statement%
                                                                 06211
      IN ['a,'z], IN ['!,'←]:  NULL;                             06212
         %punctuation, lower case alpha, upper                   06213
         case alpha, numbers%                                    06214
      =SP:                                                       06215
         BEGIN %a gap character--note its position for line
         break%                                                  06216
         IF col >= x2lim THEN EXIT;   %don't record a gap in
         last char%                                              06290
         gapcol ← col;                                           06217
         gapcc ← dgstl;                                          06218
         gapbp ← bptr;                                           06219
         END;                                                    06220
      =TAB:                                                      06221
         BEGIN %ditto%                                           06222
         gapcol ← col;                                           06223
         gapcc ← dgstl;                                          06224
         gapbp ← bptr;                                           06225
         GOTO lsfout;                                            06226
         END;                                                    06227
      =CR, =EOL:                                                 06228
         BEGIN %ditto%                                           06229
         IF col >= x2lim THEN EXIT;                              06230
         gapcol ← col;                                           06231
         gapcc ← dgstl;                                          06232
         gapbp ← bptr;                                           06233
         char ← CR;                                              06234
         GOTO lsfout;                                            06235
         END;                                                    06236
      =CA, =C.,=O, =LF, =CD, =BC, =BW, =$ascalt, IN [1B,32B],
      IN [34B,36B]: %an acceptable non-printing character%
                                                                 06237
         BEGIN                                                   06238
```

```
            gapcol ← col - hinc; %back up 1 char - we have to
            terminate current lsg before handling non-printing
            char%                                                    06239
            gapcc ← dgstl - 1;                                       06240
            gapbp ← bptr;                                            06241
            GOTO lsfout; %return the non-printing char%              06242
            END;                                                     06243
         ENDCASE %some other character - will end up displaying
         as <NP>%                                                    06244
            REPEAT CASE(CA);                                         06245
      %IF mkrflg THEN mkrset(char);% %should reinstitute marker
      display someday%                                               06246
      END;                                                           06247
   %exceeded allowed length of line segment%                         06248
   IF gapcol = 0 THEN                                                06250
      BEGIN %no gap char found%                                      06251
      gapcol ← col; %make current char be a gap char%                06252
      gapcc ← dgstl;                                                 06253
      gapbp ← bptr;                                                  06254
      IF dgstl = 0 THEN                                              06255
         char ← ENDCHR %no characters read yet%                      06256
      ELSE                                                           06257
         BEGIN                                                       06258
         rt.rtbpe ← bptr;                                            06259
         da.daccol ← col;                                            06260
         cdbckc(); %back up input one char%                          06261
         gapbp ← rt.rtbpe;                                           06262
         gapcol ← gapcol - hinc;                                     06263
         gapcc ← gapcc - 1;                                          06264
         RETURN(SP); %end of line%                                   06265
         END;                                                        06266
      END                                                            06267
   ELSE                                                              06268
      BEGIN                                                          06269
      da.daccol ← gapcol;                                            06270
      rt.rtbpe ← gapbp;                                              06271
      dgstl ← gapcc;                                                 06272
      RETURN(SP);                                                    06280
      END;                                                           06281
   (lsfout): %restore extracted variables%                           06282
   rt.rtbpe ← bptr;                                                  06283
   da.daccol ← col;                                                  06284
   RETURN(char);                                                     06285
   END.                                                              06286
                                                                     06287
(nxtlsg)   PROCEDURE (da);  %line segment termination%               05436
   %terminate current line segment and allocate a new one,           05437
   updating RT, and moving the excess characters to the              05438
   new line segment.  Allocate new LSRT if space permits.%           05439
   %-----------------%                                               05440
   LOCAL                                                             05441
      char, ls, save, oldtable;                                      05442
   REF ls, da;                                                       05443
   IF &rt >= rte THEN                                                05444
      BEGIN                                                          05445
      dismes(1, $"DISPLAY ERROR - attempting recovery...
```

```
      screen may be incomplete");                                05446
      &rt ← &rt - lsrtl; %one segment may be thrown away%         05447
      END;                                                       05448
   oldtable ← FALSE;                                             05449
   rt.rtbpe ← gapbp;                                             05450
   save ← rt.rtx2;                                               05451
   rt.rtx2 ← gapcol;                                             05452
   rt.rtnew ← TRUE;                                              05453
   &ls ← &rt := &rt + lsrtl;                                     05454
   IF &rt >= rte THEN                                            05455
      BEGIN                                                      05456
      oldtable ← dspace(10);%increase allocation 10 entries%     05457
      IF NOT oldtable THEN %failed:  either request exceeded max
      block size, or we are really out of space in the display
      block%                                                     05458
         BEGIN                                                   05459
         &rt ← &rt - lsrtl; %don't allow increment - dangerous%
                                                                 05460
         IF (lsrtl * (rtsize + 10)) < 512 %requested less than
         max% AND da.dacrow < (da.damrow - da.davinc) THEN       05461
            dismes(1, $"System had to truncate display.");      05462
         SIGNAL (spacerr, $" ");                                 05463
         END;                                                    05464
      END;                                                       05465
   dgstl ← gapcc;                                                05466
   %initialize next LSRT entry%                                  05467
      rt.rtx1 ← ls.rtx2 + ls.rthinc;                             05468
      rt.rty ← ls.rty;                                           05469
      rt.rtpartst ← TRUE;                                        05470
      rt.rtsrce ← ls.rtsrce;                                     05471
      rt.rtfont ← ls.rtfont;                                     05472
      rt.rtstid ← ls.rtstid;                                     05473
      rt.rtlev ← ls?rtlev;                                       05474
      rt.rthinc ← ls.rthinc;                                     05475
      rt.rtcsize ← ls.rtcsize;                                   05476
      rt.rtcbug ← 1;                                             05477
      rt.rtexis ← rt.rtilsg ← FALSE;                             05478
      rt.rtnew ← TRUE;                                           05479
      rt.rtbps ← rt.rtbpe ← gapbp;                               05480
      rt.rtccnt ← gapcc + 1;                                     05481
      rt.rtx2 ← save;                                            05482
      rt.rtlsid ← 0;                                             05483
   gapcol ← gapcc ← 0;                                           05484
   %now its safe to free old table%                              05485
      IF oldtable %had to reallocate% THEN                       05486
         freeblk(oldtable -  bhl, $dspblk);                      05487
   RETURN(&rt - lsrtl);                                          05488
   END.                                                          05489
                                                                 05490
(dspace) PROCEDURE(addition);                                     0942
   %allocate addition more lsrtl entries for table at rttop,
   updating globals rttop, rtsize, rt, rte%                       0943
   %returns FALSE if failed, address of old table if success,
   because some callers like to free their own table later on%
                                                                  0944
   LOCAL newtable, oldtable;                                      0945
```

```
   IF NOT (newtable ← getblk ((rtsize + addition) * lsrtl,
   &dspblk)) THEN                                                    0946
      RETURN(FALSE);                                                 0947
   %copy old to new%                                                 0948
      newtable ← newtable + bhl; %start of useable storage%          0949
      mvbfbf(rttop, newtable, rtsize * lsrtl);                       0950
      oldtable ← rttop; %need for freeing storage%                   0951
   %update global addresses%                                         0952
      rttop ← newtable;                                              0953
      &rt ← (newtable + rtsize * lsrtl);                             0954
      rte ← &rt + (addition * lsrtl);% new end + 1%                  0955
      rtsize ← rtsize + addition;                                    0956
      RETURN(oldtable);                                              0957
      END.                                                           0958
                                                                     0959
(cdbckc)   PROCEDURE;  %character output routine%                    0960
   IF rt.rtbpe.bpadr = rt.rtbps.bpadr - 1 THEN                       0961
   RETURN(ENDCHR);                                                   0962
   IF rt.rtbpe.bpbitpos <= 29 THEN                                   0963
      rt.rtbpe.bpbitpos ← rt.rtbpe.bpbitpos + 7                      0964
   ELSE                                                              0965
      BEGIN                                                          0966
      rt.rtbpe.bpbitpos ← 1;                                         0967
      BUMP DOWN rt.rtbpe.bpadr;                                      0968
      END;                                                           0969
   RETURN(.rt.rtbpe);                                                0970
   END.                                                              0971
                                                                     0972
(npstrad) PROCEDURE (npchar);%get symbol for non-printing%           0973
   %npstrad accepts a non-printing character code, npchar, as
   input, and returns the address of the appropriate global
   string used to represent that character in the display%           0974
   LOCAL index;  %for matching npcodes and npstrings%                0975
      FOR index ← 0 UP 1 UNTIL >= npsize DO                          0976
         IF npcodes[index] = npchar THEN RETURN
         (npstrings[index]);                                         0977
      RETURN (&unknown);                                             0978
   END.                                                              0979
                                                                     0980
(srch) PROCEDURE (stid, level, match, top);%find already
formatted stid%                                                      0981
   %given an STID and level, srch finds a match, if possible, in
   the LSRT, using top as the starting address of the search.
   Returns match = TRUE if found, and address of new top, if top
   may be incremented.%                                              0982
   LOCAL save;                                                       0983
   % check if top out of allowable range %                          0984
      IF (&art ←  top) >= arte OR NOT art.rtexis THEN                0985
         RETURN (FALSE,FALSE);                                       0986
   % increment table top if match on first entry%                   0987
      IF art.rtstid = stid AND art.rtlev = level THEN                0988
         BEGIN                                                       0989
         IF ((save ← lstidnxt(&art)) - lsrtl).rtpartst  AND
         (art.rty NOT= rt.rty) THEN                                  0990
            RETURN (FALSE, save) %partial stmt%                      0991
         ELSE                                                        0992
```

```
            RETURN (TRUE, save);                                    0993
         END;                                                       0994
   % main search loop%                                              0995
   &art ← lstidnxt(&art); %advance ptr to next stid%                0996
   UNTIL NOT art.rtexis OR &art >= arte OR ((save ←
   lstidnxt(&art)) - lsrtl).rtpartst AND (art.rty NOT= rt.rty))
   %partial stmt% DO                                                0997
      BEGIN                                                         0998
      IF art.rtstid = stid AND art.rtlev = level THEN %match%       0999
         BEGIN %check for special case, jumps%                      01000
         IF cdtype = dspjpf THEN RETURN (TRUE, save)                01001
         ELSE RETURN (TRUE, FALSE);                                 01002
         END;                                                       01003
      &art ← lstidnxt (&art); %advance to next stid%                01004
      END;                                                          01005
   RETURN (FALSE,FALSE); %no match found%                           01006
   END.                                                             01007
                                                                    01008
(lstidnxt) PROCEDURE (start) ; %next stid's LSRT address%           01009
   %given an lsrt entry address, return the address of the first
   entry for the statement after this one or end of table + 1%
                                                                    01010
   UNTIL start >= (arte - lsrtl) OR NOT [start].rtexis DO           01011
      IF [start].rtstid # [(start ← start + lsrtl)].rtstid THEN
                                                                    01012
         RETURN (start);                                            01013
   RETURN (start); %hit end of table - return illegal address%
                                                                    01014
   END.                                                             01015
                                                                    01016
(lscopy) PROCEDURE (option,da);%copy LSRT entries%                  01017
   %option = TRUE if only one statement to copy - else whole rest
   of &art area is copied to &rt area%                              01018
      %the big hangup with this routine is the possibility of
      running into a partial statement and having to back up%
                                                                    01019
                                                                    01020
   LOCAL oldrt, rtoffset, oldrow, lastid, oldtable;                 01021
   REF da;                                                          01022
                                                                    01023
   %check for no source data%                                       01024
      IF da.dacrow > da.damrow OR NOT art.rtexis OR &art >= arte
      THEN RETURN(FALSE);                                           01025
   %initialize locals%                                              01026
      oldrt ← &rt;                                                  01027
      oldrow ← da.dacrow;                                           01028
      lastid ← art.rtstid; %stid of first source%                   01029
      oldtable ← TRUE;                                              01030
   %segment by segment copy loop%                                   01031
      DO                                                            01032
         BEGIN                                                      01033
         lsmove(); %move one segment across to &rt%                 01034
         rt.rty ← da.dacrow;                                        01035
         IF rt.rtllsg THEN %last segment in the line%               01036
            da.dacrow := da.dacrow + da.davinc;                     01037
         &rt ← &rt + lsrtl; %increment destination%                 01038
```

```
            %check for destination overflow%                          01039
               IF &rt >= rte THEN %get more space%                    01040
                  BEGIN                                               01041
                  rtoffset ← &rt - oldrt; %offset between backup
                  factor and current destination addr%                01042
                  oldtable ← dspace(10);                              01043
                  IF NOT oldtable THEN                                01044
                     BEGIN                                            01045
                     dismes(2,$"Non-fatal System error - short on
                     space");                                         01046
                     EXIT LOOP;%allocate failed%                      01047
                     END;                                             01048
                  freeblk(oldtable - bhl, $dspblk);                   01049
                  oldrt ← &rt - rtoffset; %whole table relocated%
                                                                      01050
                  END;                                                01051
            %update backup variables in case next one incomplete%
                                                                      01052
               IF NOT /&rt-lsrtl/.rtpartst %previous seg complete%
               THEN                                                   01053
                  BEGIN                                               01054
                  oldrt ← &rt;                                        01055
                  oldrow ← da.dacrow;                                 01056
                  END;                                                01057
               END %of copy loop%                                     01058
            UNTIL (option AND art.rtstid # lastid) %done with requested
            l stmt copy%
            OR da.dacrow > da.damrow %filled screen%
            OR NOT art.rtexis OR &art >= arte %no more source data%
            OR  &rt >= rte % destination overflow%;                   01059
         IF oldtable AND
         NOT (/&rt-lsrtl/.rtpartst AND (/&rt-lsrtl/.rty NOT=
         /&art-lsrtl/.rty)) THEN RETURN(TRUE);                        01060
                                                                      01061
         %bottom of screen - back up y-cord and address to end of last
         good copy%                                                   01062
            &rt ← oldrt;                                              01063
            da.dacrow ← oldrow;                                       01064
            RETURN(FALSE);                                            01065
         END.                                                         01066
                                                                      01067
(complsg) PROCEDURE (bps1, bps2, length);  %compare two line
segments -- Return TRUE is they contain the same string of
characters%                                                           01068
   FOR length DOWN UNTIL <= 0 DO                                      01069
      IF ↑bps1 NOT= ↑bps2 THEN RETURN (FALSE);                        01070
   RETURN( TRUE );                                                    01071
   END.                                                               01072
                                                                      01073
(lsmove) PROCEDURE;                                                   01074
   %move line segment entry from &art to &rt, leaving only &art
   updated%                                                           01075
   LOCAL end, ls;                                                     01076
   REF ls;                                                            01077
   end ← (&ls ← &rt) + lsrtl;                                         01078
   DO                                                                 01079
```

```
            BEGIN                                                    01080
            ls ← art;                                                01081
            BUMP &art;                                               01082
            END                                                      01083
         UNTIL (&ls ← &ls + 1) = end;                                01084
         rt.rtnew ← FALSE;                                           01085
         RETURN END.                                                 01086
                                                                     01087
                                                                     01088
   %** leave pointers out for initial system **%                     01089
      (mkrsho) PROCEDURE;                                            01090
         RETURN;                                                     01091
         %****%                                                      01092
         END.                                                        01093
                                                                     01094
      (mkrset) PROCEDURE;                                            01095
          %col # is da.daccol %                                      01096
          %swork is current psid %                                   01097
          %ptrtb is pointer table %                                  01098
          %ptrtbl is max length of the table %                       01099
          %mkrptr is addr in ptrtb of the next possible              01100
         pointer%                                                    01101
          %ptrse2 points to loc in pointer buffer%                   01102
          % mkrend is address of end of legal pointers in table      01103
         %                                                           01104
          % mkrcnt is character number in statement of next         01105
         pointer %                                                   01106
         %-----------------%                                         01107
         %*******%                                                   01108
          RETURN;                                                    01109
         %*******%                                                   01110
         END.                                                        01111
                                                                     01112
%...display command list handling....%                              01113
   (pushdc) PROCEDURE (op, parml);                                   01114
      %enter commands into the display commands list%                 01115
      LOCAL entry,%address temp%                                     01116
         da,      %display area address%                           01117
         curblk;  %current block being filled%                      01118
      REF curblk, entry, da;                                         01119
      &curblk ← dpcmblk; %current block in use%                      01120
      &da ← curblk.cmda;                                             01121
      CASE &entry ← (curblk.cmnxt := curblk.cmnxt + 1) OF            01122
         <= curblk.cmlst:   %room in this block%                     01123
            BEGIN                                                    01124
            entry.copcode ← op;                                      01125
            entry.cparml ← parml - rttop; %make LSRT entry address
            relative in case allocations change%                     01126
            entry.ctype ← FALSE;                                     01127
            END;                                                     01128
         ENDCASE   %no room in this block%                           01129
            BEGIN                                                    01130
            IF dpcmblk ← &entry ← curblk.cbnxt %next block exists
            physically, not logically%                               01131
            THEN GO TO newblk;                                       01132
            dpcmblk ← getblk (65, &dspblk); %allocate new block%     01133
```

```
      IF NOT dpcmblk THEN                                              01134
         BEGIN                                                         01135
         dismes(1,$"FATAL DISPLAY ERROR TC and RESET
         Reenter NLS by retyping 'NLS'.");                             01136
         SIGNAL(spacerr, $"FATAL DISPLAY ERROR - type TC then
         RESET
         Reenter NLS by retyping 'NLS'"); %he's in trouble%            01137
         END;                                                          01138
      curblk.cbnxt ← &entry ← dpcmblk ← (dpcmblk + bhl); %block
      header%                                                          01139
      entry.cbprev ← &curblk;                                          01140
      entry.cmlst ← &entry + 64 - cbhl;                                01141
      entry.cbnxt ← 0;                                                 01142
      entry.cmda ← &da; %same as last block%                           01143
   (newblk):                                                           01144
      &curblk ← &entry;                                                01145
      entry.cmnxt ← &entry + cbhl;                                     01146
      &entry ← (entry.cmnxt := entry.cmnxt + 1);                       01147
      REPEAT CASE (&entry);                                            01148
      END;                                                             01149
   RETURN END.                                                         01150
                                                                       01151
(prcmds) PROCEDURE (da); %post processor%                              04565
   LOCAL mtop, wtop, dtop, mi, wi, di, ls, dls, mls, linkls,
   dlinkls, base, end, line, entry, savblk, savnxt, writend, curblk,
   clear, replace, noop, ms[150], ws[150], ds[150];                    04566
   REF art, da, ls, dls, mls, entry, savblk, linkls, dlinkls;          04567
   %initialization%                                                    04568
   curblk ← dpcmblk; %last block written%                              04569
   &entry ← [curblk].cmnxt; %addr + 1 of last used entry%              04570
   %main processing loop%                                              04571
   CASE nldevice OF                                                    04572
      =devlproc: %alpha-numeric display%                               04573
         BEGIN                                                         04574
         %initialize%                                                  04575
            clear ← 100; replace ← 101; noop ← 102;                    04576
               %internal values for copcode%                           04577
            mtop ← $ms; wtop ← $ws; dtop ← $ds;                        04578
         UNTIL NOT cnxtaddress(&entry, curblk, backward  :&entry,
         curblk) DO                                                    04579
            CASE entry.copcode OF                                      04580
               =writelsg:                                              04581
                  BEGIN                                                04582
                  &ls ← entry.cparm1 + rttop;                          04583
                  IF ls.rtnew THEN                                     04584
                     [wtop := wtop + 1] ← &entry;                      04585
                  END;                                                 04586
               =movelsg: [mtop := mtop + 1] ← &entry;                  04587
               =deletlsg: [dtop := dtop + 1] ← &entry;                 04588
               =blankda:  %process now%                                04589
                  clearda(&da);                                        04590
               =suppressda: %process now%                              04591
                  supda(&da);                                          04592
               =restoreda: %process now%                               04593
                  resda(&da);                                          04594
               ENDCASE err($"undefined display command");              04595
```

```
(prcmd1):                                                    04596
%check for several writes/deletes/moves to same line --
link them together%                                          04597
   FOR wi ← wtop - 1 DOWN UNTIL < $ws DO                     04598
      BEGIN                                                  04599
      &ls ← [[wi]].cparm1 + rttop;                           04600
      IF NOT ls.rt1plink AND NOT ls.rt11sg #then %look for
      others on same line%                                   04601
         BEGIN                                               04602
         line ← ls.rty;                                      04603
         &dls ← &ls;                                         04604
         FOR mi ← wi - 1 DOWN UNTIL < $ws DO                 04605
            BEGIN                                            04606
            &mls ← [[mi]].cparm1 + rttop;                    04607
            IF mls.rty = line THEN                           04608
               BEGIN                                         04609
               dls.rt1plink ← &mls - &dls;                   04610
               &dls ← &mls;                                  04611
               [[mi]].copcode ← noop;                        04612
               END;                                          04613
            END;                                             04614
         END;                                                04615
      END;                                                   04616
   FOR di ← dtop - 1 DOWN UNTIL < $ds DO                     04617
      BEGIN                                                  04618
      &ls ← [[di]].cparm1 + rttop;                           04619
      IF NOT ls.rt1plink AND NOT ls.rt11sg THEN %look for
      others on same line%                                   04620
         BEGIN                                               04621
         line ← ls.rty;                                      04622
         &dls ← &ls;                                         04623
         FOR mi ← di - 1 DOWN UNTIL < $ds DO                 04624
            BEGIN                                            04625
            &mls ← [[mi]].cparm1 + rttop;                    04626
            IF mls.rty = line THEN                           04627
               BEGIN                                         04628
               dls.rt1plink ← &mls - &dls;                   04629
               &dls ← &mls;                                  04630
               [[mi]].copcode ← noop;                        04631
               END;                                          04632
            END;                                             04633
         END;                                                04634
      END;                                                   04635
   FOR mi ← mtop - 1 DOWN UNTIL < $ms DO                     04636
      BEGIN                                                  04637
      &ls ← [[mi]].cparm1 + rttop;                           04638
      IF NOT ls.rt1plink AND NOT ls.rt11sg THEN %look for
      others on same line%                                   04639
         BEGIN                                               04640
         line ← ls.rty;                                      04641
         &dls ← &ls;                                         04642
         FOR wi ← mi - 1 DOWN UNTIL < $ms DO                 04643
            BEGIN                                            04644
            &mls ← [[wi]].cparm1 + rttop;                    04645
            IF mls.rty = line THEN                           04646
               BEGIN                                         04647
```

```
                    dls.rtlplink ← &mls - &dls;                    04648
                    &dls ← &mls;                                   04649
                    [[wi]].copcode ← noop;                         04650
                    END;                                           04651
                 END;                                              04652
              END;                                                 04653
           END;                                                    04654
        (prcmd2):                                                  04655
        CASE (IF mtop = $ms THEN 0 ELSE 4) .V (IF wtop = $ws THEN 0
        ELSE 2) .V (IF dtop = $ds THEN 0 ELSE 1) OF                04656
           = 1: %deletes only - no moves or writes%                04657
              BEGIN %process all deletes as clears%                04658
              (prcmd3):                                            04659
              FOR di ← $ds UP UNTIL >= dtop DO                     04660
                 IF [[di]].copcode = deletlsg THEN                 04661
                    BEGIN                                          04662
                    &dls ← &ls ← [[di]].cparm1 + rttop;            04663
                    LOOP                                           04664
                       BEGIN                                       04665
                       dls.rtlsid ← 0;                             04666
                       IF dls.rtlplink THEN &dls ← &dls +
                       dls.rtlplink                                04667
                       ELSE EXIT LOOP;                             04668
                       END;                                        04669
                    cline(ls.rtx1+da.daleft, ls.rty+da.datop,
                    (dls.rtx2-ls.rtx1)/ls.rthinc + 1);             04670
                    END;                                           04671
              END;                                                 04672
           = 2: %writes only - no moves or deletes%                04673
              BEGIN                                                04674
              (prcmd4):                                            04675
              % process writes as replaces%                        04676
              FOR wi ← wtop-1 DOWN UNTIL < $ws DO                  04677
                 IF [[wi]].copcode = writelsg THEN                 04678
                    BEGIN                                          04679
                    &ls ← [[wi]].cparm1 + rttop;                   04680
                    IF ls.rtx1 > 0  AND (lptype .A 17B) = deltadata
                    THEN %must pad for display delay%              04681
                       cline(da.daleft, ls.rty+da.datop,
                       ls.rtx1/ls.rthinc);                         04682
                    wrtstr(&da, &ls);                              04683
                    ls.rtlsid ← da.dahandle := da.dahandle+1;      04684
                    WHILE ls.rtlplink DO                           04685
                       BEGIN                                       04686
                       &ls ← &ls + ls.rtlplink;                    04687
                       ls.rtlsid ← da.dahandle := da.dahandle+1;
                                                                   04688
                       END;                                        04689
                    END;                                           04690
              END;                                                 04691
           IN [3,7]: % writes, deletes, and moves%                 04692
              BEGIN                                                04693
              (prcmd5):                                            04694
              % For each write, if there is a delete for the same
              line  and no moves FROM/TO above it then make the
              write a replace and the delete a clear (if old line
```

```
is farther to left than new line) otherwise a noop.
Otherwise, process writes as inserts and deletes as
deletes%                                                    04695
IF dtop > $ds THEN                                          04696
   BEGIN                                                    04697
   FOR wi ← wtop-1 DOWN UNTIL < $ws DO                      04698
      IF [[wi]].copcode = writelsg THEN                     04699
         BEGIN                                              04700
         &ls ← [[wi]].cparm1 + rttop;                       04701
         (prcmd6):                                          04702
         ls.rtlsid ← ls.rtx2; %save rtx2 -- it may
         get smashed%                                       04703
         FOR di ← $ds UP UNTIL >= dtop DO                   04704
            IF [[d ]].copcode = deletlsg THEN               04705
               BEGIN                                        04706
               &dls ← [[di]].cparm1 + rttop;                04707
               (prcmd7):                                    04708
               IF ls.rty = dls.rty THEN %there is a
               write and delete to same line%               04709
                  BEGIN                                     04710
                  end ← TRUE;                               04711
                  %determine if any moves to/from
                  above this line%                          04712
                     arte ← oldlsrt + (lsrtl *
                     da.dalsz);                             04713
                     FOR mi ← $ms UP UNTIL >= mtop DO
                                                            04714
                        IF [[mi]].copcode = movelsg
                        THEN                                04715
                           BEGIN                            04716
                           &mls ← [[mi]].cparm1 +
                           rttop;                           04717
                           IF mls.rty <= ls.rty THEN
                                                            04718
                           BEGIN %a line is being
                           moved to above this one,
                           thus delete must delete
                           the line and write must
                           insert new line%                 04719
                           end ← FALSE;                     04720
                           EXIT;                            04721
                           END;                             04722
                           %otherwise must see if
                           moving from above this
                           line%                            04723
                           FOR &art ← oldlsrt UP
                           lsrtl UNTIL >= arte DO
                                                            04724
                           IF art.rtexis AND
                           art.rtlsid = mls.rtlsid
                           THEN                             04725
                           EXIT;                            04726
                           IF &art < arte AND art.rty
                           <= ls.rty THEN                   04727
                           BEGIN %a line is being
                           moved from above this one,
```

```
                                thus delete must delete
                                the line and write must
                                insert new line%        04728
                                end ← FALSE;            04729
                                EXIT;                   04730
                                END;                    04731
                                END;                    04732
                          IF end THEN %no moves to/from above
                          this line%                    04733
                             BEGIN                      04734
                             (prcmd8):                  04735
                             [[wi]].copcode ← replace;  04736
                             [[di]].copcode ← noop;     04737
                             %check for need to write
                             trailing blanks%           04738
                                &dlinkls ← &dls;        04739
                                IF dls.rtlplink THEN    04740
                                   WHILE dlinkls.rtlplink DO
                                   &dlinkls ← &dlinkls +
                                   dlinkls.rtlplink;    04741
                                &linkls ← &ls;          04742
                                IF ls.rtlplink THEN     04743
                                   WHILE linkls.rtlplink DO
                                   &linkls ← &linkls +
                                   linkls.rtlplink;     04744
                                IF dlinkls.rtx2 > linkls.rtx2
                                THEN % must write trailing
                                blanks%                 04745
                                   linkls.rtx2 ←
                                   dlinkls.rtx2;        04746
                             IF dls.rtxl < ls.rtxl THEN %must
                             write begining blanks%     04747
                                BEGIN                   04748
                                [[di]].copcode ← clear; 04749
                                dls.rtx2 ← ls.rtxl;     04750
                                END;                    04751
                             END;                       04752
                          END;                          04753
                       END;                             04754
                 END;                                   04755
           END;                                         04756
%moves -- erase old instances of these line segments%
                                                        04757
        FOR mi ← mtop-1 DOWN UNTIL < $ms DO             04758
           IF [[mi]].copcode = movelsg THEN             04759
              BEGIN                                     04760
              (prcm30):                                 04761
              &mls ← [[mi]].cparml + rttop;             04762
              %determine if moved up or down correct
              number of lines lines%                    04763
                 arte ← oldlsrt + (lsrtl * da.dalsz);   04764
                 FOR &art ← oldlsrt UP lsrtl UNTIL >= arte
                 DO                                     04765
                    IF art.rtexis AND art.rtlsid =
                    mls.rtlsid THEN                     04766
                       EXIT;                            04767
```

```
                    (prcm31):                                   04768
                    IF &art < arte THEN                         04769
                       BEGIN                                    04770
                       line ← [[mi]].cparm2 ← art.rty;
                       %y[old]%                                 04771
                       IF NOT da.dalneighbor AND NOT
                       da.darneighbor AND NOT da.datneighbor
                       AND NOT da.dabneighbor THEN              04772
                          BEGIN                                 04773
                          (prcm32):                             04774
                          FOR di ← $ds UP UNTIL >= dtop DO
                                                                04775
                             IF [[di]].copcode = deletlsg
                             THEN                               04776
                                BEGIN                           04777
                                &dls ← [[di]].cparm1 + rttop;
                                                                04778
                                IF dls.rty < line THEN %line
                                is moving up%                   04779
                                   BEGIN                        04780
                                   line ← line - da.davinc;
                                                                04781
                                   END;                         04782
                                END;                            04783
                          FOR wi ← wtop-1 DOWN UNTIL < $ws DO
                                                                04784
                             IF [[wi]].copcode = writelsg
                             THEN                               04785
                                BEGIN                           04786
                                &ls ← [[wi]].cparm1 + rttop;
                                                                04787
                                IF ls.rty <= line THEN          04788
                                   BEGIN %line is moving
                                   down%                        04789
                                   line ← line + da.davinc;
                                                                04790
                                   END;                         04791
                                END;                            04792
                          IF line = mls.rty THEN
                          [[mi]].copcode ← noop;                04793
                          END;                                  04794
                       END                                      04795
                    ELSE %what to do???% [[mi]].copcode ←
                    noop;                                       04796
                 END;                                           04797
              %clear lines where line segments were%            04798
                 FOR mi ← mtop-1 DOWN UNTIL < $ms DO            04799
                    IF [[mi]].copcode = movelsg THEN            04800
                       BEGIN                                    04801
                       &ls ← [[mi]].cparm1 + rttop;             04802
                       (prcm33):                                04803
                       % dc a cline where this line segment
                       was.%                                    04804
                          cline(da.daleft,
                          [[mi]].cparm2+da.datop,
                          (da.daright-da.daleft)/da.dahinc);
```

```
                                                                    04805
            END;                                                    04806
%clears%                                                            04807
   FOR di ← $ds UP UNTIL >= dtop DO                                 04808
      IF [[di]].copcode = clear THEN                                04809
         BEGIN                                                      04810
         &ls ← [[di]].cparm1 + rttop;                               04811
         LOOP                                                       04812
            BEGIN                                                   04813
            (prcmd9):                                               04814
            ls.rtlsid ← 0;                                          04815
            cline(ls.rtx1+da.daleft, ls.rty+da.datop,
            (ls.rtx2-ls.rtx1)/ls.rthinc + 1);                       04816
            IF ls.rtlplink THEN                                     04817
               &ls ← &ls + ls.rtlplink                              04818
            ELSE EXIT LOOP;                                         04819
            END;                                                    04820
         END;                                                       04821
%replaces%                                                          04822
   FOR wi ← wtop-1 DOWN UNTIL < $ws DO                              04823
      IF [[wi]].copcode = replace THEN                              04824
         BEGIN                                                      04825
         &ls ← [[wi]].cparm1 + rttop;                               04826
         (prcm11):                                                  04827
         IF ls.rtx1 > 0 AND (lptype .A 17B) =
         deltadata THEN %must pad for display delay%
                                                                    04828
            cline(da.daleft, ls.rty+da.datop,
            ls.rtx1);                                               04829
         wrtstr(&da, &ls);                                          04830
         ls.rtx2 ← ls.rtlsid; %restore rtx2%                        04831
         ls.rtlsid ← da.dahandle := da.dahandle+1;
                                                                    04832
         WHILE ls.rtlplink DO                                       04833
            BEGIN                                                   04834
            &ls ← &ls + ls.rtlplink;                                04835
            ls.rtlsid ← da.dahandle := da.dahandle+1;
                                                                    04836
            END;                                                    04837
         END;                                                       04838
%deletes%                                                           04839
   FOR di ← $ds UP UNTIL >= dtop DO                                 04840
      IF [[di]].copcode = deletlsg THEN                             04841
         BEGIN                                                      04842
         &ls ← [[di]].cparm1 + rttop;                               04843
         (prcm12):                                                  04844
         ls.rtlsid ← 0;                                             04845
         IF da.dalneighbor OR da.darneighbor OR
         da.datneighbor OR da.dabneighbor THEN %has a
         left, right , top, or bottomneighbor window%
                                                                    04846
            LOOP                                                    04847
               BEGIN                                                04848
               (prcm13):                                            04849
               [[di]].copcode ← clear;                              04850
               cline(da.daleft+ls.rtx1,
```

```
                    da.datop+ls.rty,
                    ls.rtx2-ls.rtx1+ls.rthinc);          04851
                    IF ls.rtlplink THEN                  04852
                       &ls ← &ls + ls.rtlplink           04853
                    ELSE EXIT LOOP;                      04854
                    END                                  04855
                ELSE                                     04856
                   dline(da.daleft, ls.rty+da.datop);    04857
                END;                                     04858
%inserts%                                                04859
   FOR wi ← wtop-1 DOWN UNTIL < $ws DO                   04860
      IF [[wi]].copcode = writelsg THEN                  04861
         BEGIN                                           04862
         &ls ← [[wi]].cparm1 + rttop;                    04863
         (prcm14):                                       04864
         IF da.daineighbor OR da.darneighbor OR
         da.dabneighbor OR da.datneighbor THEN           04865
            BEGIN                                        04866
            (prcm15):                                    04867
            [[wi]].copcode ← replace;                    04868
            &ls ← [[wi]].cparm1 + rttop;                 04869
            cline(da.daleft, da.datop+ls.rty,
            da.daright-da.daleft);                       04870
            END                                          04871
         ELSE                                            04872
            BEGIN                                        04873
            (prcm16):                                    04874
            IF ls.rty + da.datop = lpymax THEN           04875
               cline(da.daleft, ls.rty+da.datop,
               da.daright-da.daleft)                     04876
                  %can't do insert on last line on
                  screen %                               04877
            ELSE %insert new line%                       04878
               inline(da.daleft, ls.rty+da.datop,
               $"", FALSE);                              04879
            IF ls.rtx1 > 0 AND (lptype .A 17B) =
            deltadata THEN %must pad for display
            delay%                                       04880
               cline(da.daleft, ls.rty+da.datop,
               ls.rtx1);                                 04881
            END;                                         04882
         wrtstr(&da, &ls);                               04883
         ls.rtlsid ← da.dahandle := da.dahandle+1;
                                                         04884
         WHILE ls.rtlplink DO                            04885
            BEGIN                                        04886
            &ls ← &ls + ls.rtlplink;                     04887
            ls.rtlsid ← da.dahandle := da.dahandle+1;
                                                         04888
            END;                                         04889
         END;                                            04890
%moves%                                                  04891
   %write them where they are supposed to be%            04892
      FOR mi ← mtop-1 DOWN UNTIL < $ms DO                04893
         IF [[mi]].copcode = movelsg THEN                04894
            BEGIN                                        04895
```

```
                        &ls ← [[mi]].cparm1 + rttop;                    04896
                        (prcm21):                                       04897
                        % do a cline where this line segment is
                        supposed to be.%                                04898
                           cline(da.daleft, ls.rty+da.datop,
                           (da.daright-da.daleft)/ls.rthinc);
                                                                        04899
                        %fix up byte pointers if necessary%             04900
                           &dls ← &ls;                                  04901
                           LOOP                                         04902
                              BEGIN                                     04903
                              (prcm22):                                 04904
                              IF NOT dls.rtstid.stastr THEN %must
                              get SDB%                                  04905
                                 BEGIN                                  04906
                                 lodsdb(getsdb(dls.rtstid) :
                                 base);                                 04907
                                 base ← base + sdbhdl;                  04908
                                    %add in length of sdb header
                                    to get base of char string%
                                                                        04909
                                 end ← dls.rtbpe.bpadr -
                                 dls.rtbps.bpadr;                       04910
                                    %number of words between
                                    start and end%                      04911
                                 dls.rtbps ← stbptr(dls.rtccnt -
                                 1) + base;                             04912
                                 dls.rtbpe.bpadr ←
                                 dls.rtbps.bpadr + end;                 04913
                                 IF dls.rtlplink THEN &dls ← &dls
                                 + dls.rtlplink                         04914
                                 ELSE EXIT LOOP;                        04915
                                 END;                                   04916
                              END:                                      04917
                        wrtstr(&da, &ls);                               04918
                        END:                                            04919
                                                                        04920
           END;
        ENDCASE; %no moves, writes, or deletes%                         04921
     END; %display device%                                              04922
ENDCASE                                                                 04923
     BEGIN                                                              04924
     UNTIL NOT cnxtaddress(&entry, curblk, backward  :&entry,
     curblk ) DO                                                        04925
        BEGIN                                                           04926
        CASE entry.copcode OF                                           04927
           =writelsg:                                                   04928
              BEGIN                                                     04929
              writend ← &entry; %last entry to process%                 04930
              UNTIL NOT cnxtaddress (&entry, curblk, backward
              :&entry, curblk) OR entry.copcode # writelsg DO
              NULL;                                                     04931
              %mark start of writes for later%                          04932
              savnxt ← &entry;                                          04933
              &savblk ← curblk;                                         04934
              %process writes in reverse order%                         04935
              DO                                                        04936
```

```
                    BEGIN                                                   04937
                    &ls ← entry.cparml + rttop;                             04938
                    IF ls.rtnew AND NOT ls.rtlsid THEN                      04939
                       wrtstr(&da, &ls)                                     04940
                    END                                                     04941
                 UNTIL NOT cnxtaddress (&entry, curblk, forward
                 :&entry, curblk) OR &entry = writend + 1;                  04942
                 &entry ← savnxt;                                           04943
                 curblk ← &savblk;                                          04944
                 END;                                                       04945
              =movelsg:                                                     04946
                 BEGIN                                                      04947
                 &ls ← entry.cparml + rttop;                                04948
                 ls.rtbps ← -1; %old string%                                04949
                 ls.rtbpe ← 0;                                              04950
                 IF ls.rty = da.dabottom - da.datop THEN                    04951
                    ls.rty ← ls.rty - 256;                                  04952
                    %if a cord is zero, the strda jsys assumes you
                    do not want that cord changed--see wrtstr and
                    djsys documentation for strda%                          04953
                    wrtstr(&da, &ls);                                       04954
                    END;                                                    04955
              =deletlsg:                                                    04956
                 BEGIN                                                      04957
                 &ls ← entry.cparml + rttop;                                04958
                 ls.rtbps ← ls.rtbpe ← 0; %delete%                          04959
                 wrtstr(&da, &ls);                                          04960
                 END;                                                       04961
              =blankda:                                                     04962
                 BEGIN                                                      04963
                 clearda(&da);                                              04964
                 END;                                                       04965
              =suppressda:                                                  04966
                 supda(&da);                                                04967
              =restoreda:                                                   04968
                 resda(&da);                                                04969
              ENDCASE err($"System error:  undefined display
              command...
              ↑C and RESET");                                               04970
           END; %of main command processing loop%                           04971
        END; %display device%                                               04972
%deallocate all but first command storage block%                            04973
cmdfree(dpcmblk);                                                           04974
%free any outstanding signature or statement number strings -
this code not currently in use - being freed in daupdate and
clrall%
%IF NOT da.davspec.vsindf AND NOT da.davspec.vsstnf THEN RETURN;
                                                                            04976
end ← da.dalsz * lsrtl + da.dalsrt;                                         04977
&ls ← da.dalsrt;                                                            04978
DO                                                                          04979
   BEGIN                                                                    04980
   IF NOT ls.rtexis THEN RETURN;                                            04981
   CASE ls.rtsrce OF                                                        04982
      =srcsig, =srcstno:                                                    04983
         <STGMGT, freestring>(ls.rtbps.bpadr, $dsppblk);                    04984
```

```
      ENDCASE NULL;                                                    04985
      END                                                              04986
   UNTIL (&ls ← &ls + lsrtl) >= end;%                                  04987
   RETURN END.                                                         04988
                                                                       04989
(wrtstr) PROCEDURE (da, ls); %write string in display area%            04990
   %write a string onto the display.  Uses the information in the
   display area and line segment records to determiine what to write
   and where and how.  If the rtbps field is -1 then the old string
   is used.%                                                           04991
   %-----------------%                                                 04992
   LOCAL nls, store, savestart, save, startbp, endbp, i, char, l,
   mx, blanks, slength, bp, length, format;                            04993
   LOCAL STRING send[200];                                             04994
   REF da, ls, nls;                                                    04995
   CASE nldevice OF                                                    04996
      = devlproc:                                                      04997
         BEGIN                                                         04998
         %send position cursor code%                                   04999
            position( ls.rtxl+da.daleft, ls.rty+da.datop );            05000
         %send text%                                                   05001
            LOOP                                                       05002
               BEGIN                                                   05003
               startbp ← ls.rtbps;                                     05004
               endbp ← ls.rtbpe;                                       05005
               save ← MIN(da.daright, ls.rtx2+da.daleft);              05006
                  %rtx2 points to the last char of the string%         05007
               savestart ← store ← chbmty + $send;                     05008
               *send* ← NULL;                                          05009
               l ← send.L;  mx ← send.M;                               05010
               blanks ← 0;                                             05011
               i ← ls.rtxl+da.daleft;                                  05012
               FOR i UP ls.rthinc UNTIL > save DO                      05013
                  BEGIN                                                05014
                  IF startbp = endbp THEN %fill with blanks%           05015
                     BUMP blanks                                       05016
                  ELSE                                                 05017
                     BEGIN                                             05018
                     CASE (char ← ↑startbp) OF                         05019
                        = TAB: char ← lptab;                           05020
                        = EOL: char ← CR;                              05021
                        ENDCASE;                                       05022
                     %*send* ← *send*, char;%                          05023
                        ↑store ← char;                                 05024
                        BUMP l;                                        05025
                     END;                                              05026
                  IF l >= mx THEN %time to sout -- string full%        05027
                     BEGIN                                             05028
                     !sout(dspjfn, savestart, -1);                     05029
                     IF ldspjfn THEN                                   05030
                        !sout(ldspjfn, savestart, -1);                 05031
                     store ← savestart;                                05032
                     *send* ← NULL;                                    05033
                     l ← send.L;                                       05034
                     END;                                              05035
                  END;                                                 05036
```

```
            IF 1 THEN %sout the string%                                  05037
               BEGIN                                                     05038
               !sout(dspjfn, savestart, -1);                             05039
               IF ldspjfn THEN                                           05040
                  !sout(ldspjfn, savestart, -1);                         05041
               store ← savestart;                                        05042
               *send* ← NULL;                                            05043
               l ← send.L;                                               05044
               END;                                                      05045
            IF blanks IN [1, lpxmax] THEN                                05046
               cline(current, current, blanks);                          05047
            %check to see if several line segments are to be
            writtten on this line%                                       05048
               IF ls.rtlplink THEN  %there is another line
               segment -- loop again%                                    05049
                  BEGIN                                                  05050
                  &nls ← &ls + ls.rtlplink;                              05051
                  blanks ← nls.rtxl - ls.rtx2 -1;                        05052
                  IF blanks IN ?(1, lpxmax/ THEN                          05053
                     cline(current, current, blanks);                    05054
                  &ls ← &nls;                                            05055
                  REPEAT LOOP;                                           05056
                  END                                                    05057
               ELSE EXIT LOOP;                                           05058
            END;                                                         05059
         track();                                                        05060
         END;                                                            05061
      = imlac0, = imlac1:                                                05062
         BEGIN                                                           05063
         CASE ls.rtbps OF                                                05064
            = 0: %delete string%                                         05065
               BEGIN                                                     05066
               IF ls.rtlsid = 0 OR NOT chkbit(da.dalsidb, ls.rtlsid,
               lsbtsize) THEN BEGIN                                      05067
                  dismes(2, $"wrtstr: suppressing a non-existant
                  string");                                              05068
                  RETURN;                                                05069
                  END;                                                   05070
               *send* ← begmsg, 5+remfudge, remssda,
               da.dahandle.daidr1 + remfudge,
               da.dahandle.daidr2 + remfudge, ls.rtlsid, TRUE;           05071
               !sout(dspjfn, chbmty + $send ,-send.L);                   05072
               dassnbit(da.dalsidb,ls.rtlsid,lsbtsize);                  05073
               RETURN;                                                   05074
               END;                                                      05075
            = -1: %use old string%                                       05076
               IF da.daseq THEN RETURN                                   05077
               ELSE slength ← -1;                                        05078
            ENDCASE %new string%                                         05079
               BEGIN %get string length%                                 05080
               startbp ← ls.rtbps;                                       05081
               endbp ← ls.rtbpe;                                         05082
               slength ←slngth(startbp, endbp);                          05083
               END;                                                      05084
         %setup format info and assign or check stid
         if default values, use default constants %                      05085
```

```
format ← 0;                                                          05086
save ← (da.dabottom-da.datop-ls.rty)/256;                            05087
IF ls.rtlsid = 0 THEN  %new string%                                  05088
   BEGIN                                                             05089
   ls.rtlsid ←  assnbit(da.dalsidb,lsbtsize);                        05090
   format.xyds ← FALSE;                                              05091
   END                                                               05092
ELSE                                                                 05093
   BEGIN                                                             05094
   IF NOT chkbit(da.dalsidb,ls.rtlsid,lsbtsize) THEN                 05095
      BEGIN                                                          05096
      dismes(2,$"wrtstr: lsbitable fouled - CHKBIT
      failure!");                                                    05097
      RETURN;                                                        05098
      END;                                                           05099
   format.xyds ← IF ls.rtxl OR save THEN FALSE ELSE TRUE;
                                                                     05100
   END;                                                              05101
IF ls.rtfont = da.dafont THEN format.fdd ← TRUE                      05102
ELSE format.fdd ← format.fds ← FALSE;                                05103
IF ls.rtcsize = da.dacsize THEN format.sdd ← TRUE                    05104
ELSE format.sdd ← format.sds ← FALSE;                                05105
IF ls.rthinc = da.dahinc THEN format.idd ← TRUE                      05106
ELSE format.idd ← format.ids ← FALSE;                                05107
%send command parameters plus string%                                05108
   length ← slength ←                                                05109
      MIN((da.daright-ls.rtxl)/ls.rthinc, slength); %safety
      precausion%                                                    05110
   IF da.daseq THEN % if sequential then use APPEND
   command%                                                          05111
      BEGIN                                                          05112
      *send* ← begmsg, length+3+remfudge, apsda,
        da.dahandle.daidr1+remfudge,
      da.dahandle.daidr2+remfudge;                                   05113
      bp ← chbptr(send.L) + $send;                                   05114
      END                                                            05115
   ELSE                                                              05116
      BEGIN                                                          05117
      %compute message length%                                       05118
      length ← MAX(0, length) + 6 + remfudge;                        05119
         IF NOT format.xyds THEN length ← length + 4;                05120
         IF NOT format.sds AND NOT format.sdd THEN BUMP
         length;                                                     05121
         IF NOT format.ids AND NOT format.idd THEN BUMP
         length;                                                     05122
         IF NOT format.fds AND NOT format.fdd THEN BUMP
         length;                                                     05123
      %string manipulation as per (JOURNAL, 11726,)%                 05124
         *send* ← begmsg, length,  nwstrda,
          da.dahandle.daidr1+remfudge,
         da.dahandle.daidr2+remfudge, ls.rtlsid,
          slength + 1, format + remfudge;                            05125
         bp ← chbptr(send.L) + $send;                                05126
      %cordinates%                                                   05127
         IF NOT format.xyds THEN                                     05128
            BEGIN                                                    05129
```

```
                ↑bp ← remfudge + (ls.rtx1/256).xc1;                    05130
                ↑bp ← remfudge + (ls.rtx1/256).xc2;                    05131
                ↑bp ← remfudge + save.yc1;                             05132
                ↑bp ← remfudge + save.yc2;                             05133
                send.L ← send.L + 4;                                   05134
                END;                                                   05135
            %character size%                                           05136
               IF NOT format.sds AND NOT format.sdd THEN               05137
                BEGIN                                                  05138
                ↑bp ← ls.rtcsize;                                      05139
                BUMP send.L;                                           05140
                END;                                                   05141
            %horizontal increment%                                     05142
               IF NOT format.ids AND NOT format.idd THEN               05143
                BEGIN                                                  05144
                ↑bp ← (ls.rthinc/128) + 1;                             05145
                BUMP send.L;                                           05146
                END;                                                   05147
            %font%                                                     05148
               IF NOT format.fds AND NOT format.fdd THEN               05149
                BEGIN                                                  05150
                ↑bp ← ls.rtfont*2 + 1;                                 05151
                BUMP send.L;                                           05152
                END;                                                   05153
            END;                                                       05154
         %now send header and string%                                  05155
         IF slength > 0 THEN                                           05156
            UNTIL startbp = endbp                                      05157
               OR startbp.bpadr > endbp.bpadr                          05158
               OR (send.L ← send.L +1) >= send.M DO                    05159
               ↑bp ← IF (char ← ↑startbp) = EOL THEN CR ELSE
               char;                                                   05160
         !sout(dspjfn, chbmty + $send, -send.L) ;                      05161
      END;                                                             05162
   ENDCASE                                                             05163
      BEGIN                                                            05164
      r1.sdaid ← da.dahandle;                                          05165
      r1.sstrid ← ls.rtlsid;                                           05166
      save ← r1;                                                       05167
      r2 ← ls.rtbps;                                                   05168
      r3 ← ls.rtbpe;                                                   05169
      r4.sxcord ← ls.rtx1/256;                                         05170
      r4.sycord ← ((da.dabottom-da.datop) - ls.rty)/256;               05171
      %if default values, use default constants%                       05172
      IF ls.rtfont = da.dafont THEN r4.sfont ← -1                      05173
      ELSE r4.sfont ← ls.rtfont*2 + 1;                                 05174
      IF ls.rtcsize = da.dacsize THEN r4.scsize ← -1                   05175
      ELSE r4.scsize ← ls.rtcsize*2 + 1;                               05176
      IF ls.rthinc = da.dacsize THEN r4.shinc ← -1                     05177
      ELSE r4.shinc ← (ls.rthinc/128) + 1;                             05178
      IF NOT SKIP !JSYS strda THEN                                     05179
         IF r1 = syserr THEN NULL                                      05180
         ELSE                                                          05181
            dismes(2,$"NLS Display Error")                             05182
      ELSE ls.rtlsid ← r1 .A 18M;                                      05183
      IF da.dalhandle THEN %linked display area%                       05184
```

```
            BEGIN                                                   05185
            r1 ← save;                                              05186
            r1.sdaid ← da.dalhandle;                                05187
            IF NOT SKIP !JSYS strda THEN                            05188
               IF r1 = syserr THEN NULL                             05189
               ELSE dismes(2, $"Link Display Error");               05190
            END;                                                    05191
         END;                                                       05192
   RETURN;                                                          05193
   END.
                                                                    05194
(cmdfree) PROCEDURE (blkadress); %free command blocks%              01772
   %starting at blkadress, free all display command blocks but the
   original one (which has no previous block address).%             01773
   LOCAL savblk, curblk;                                            01774
   REF savblk;                                                      01775
   &savblk ← blkadress; %original starting block--last used%        01776
   UNTIL NOT $savblk.cbprev DO                                      01777
      BEGIN                                                         01778
      curblk ← savblk.cbprev;                                       01779
      freeblk(&savblk - bhl, $dspblk);                              01780
      &savblk ← curblk;                                             01781
      END;                                                          01782
      dpcmblk ← $commands; %reset global block pointer%             01783
      RETURN END.                                                   01784
                                                                    01785
(cnxtaddress) PROCEDURE (entry, block, direction);                  01786
   %updates entry and block in direction specified%                 01787
   REF block;                                                       01788
   CASE direction OF                                                01789
      =forward:                                                     01790
         BEGIN                                                      01791
         BUMP entry;                                                01792
         IF entry <= block.cmnxt - 1 THEN RETURN(TRUE, entry,
         &block);                                                   01793
         BUMP DOWN entry; %restore previous value%                  01794
         IF NOT block.cbnxt %no new block%                          01795
            THEN RETURN(FALSE, entry, &block);                      01796
         entry ← (&block ← block.cbnxt) + cbhl;                     01797
         RETURN(TRUE, entry, &block);                               01798
         END;                                                       01799
      =backward:                                                    01800
         BEGIN                                                      01801
         BUMP DOWN entry;                                           01802
         IF entry >= &block + cbhl THEN RETURN(TRUE, entry, &block);
                                                                    01803
         BUMP entry; %restore previous value%                       01804
         IF NOT block.cbprev THEN RETURN(FALSE, entry, &block);     01805
         &block ← block.cbprev;                                     01806
         entry ← block.cmlst; %last used%                           01807
         RETURN(TRUE, entry, &block);                               01808
         END;                                                       01809
      ENDCASE err($"FATAL DISPLAY SYSTEM ERROR...
      Type ↑C and RESET");                                          01810
      END.                                                          01811
                                                                    01812
```

```
    (cmdinit) PROCEDURE(block, da); %initialize display command block%
                                                                    01813
       REF block;                                                   01814
       dpcmblk ← &block; %global current block%                     01815
       block.cbprev ← block.cbnxt ← 0;                              01816
       block.cmlst ← &block + 64 - cbhl;                            01817
       block.cmnxt ← &block + cbhl;                                 01818
       block.cmda ← da;                                             01819
       RETURN END.                                                  01820
                                                                    01821
%...clearing da's, tables ....%                                     01822
    (clrall) PROCEDURE (da, free);  %reset LSRT entries for the da% 01823
       %clrall has three functions
       --zeros the LSRT entries for the given display
       --frees storage used for statement numbers and signatures
       --reallocates LSRT space if it's not being used              01824
       In general, if da=0 then these happen for all random display
       areas.  If free = TRUE, free strings.%                       01825
       %----------------%                                           01826
       LOCAL                                                        01827
          end,   %end of LSRT table plus one entry%                 01828
          index,                                                    01829
          straddr,  %signature or stmt # string address%            01830
          daend,                                                    01831
          used,  %number of LSRT words actually in use%             01832
          oldtable,                                                 01833
          oldsize,                                                  01834
          count,                                                    01835
          lines;                                                    01836
       REF da, index;                                               01837
       IF NOT &da THEN                                              01838
          daend ← (&da ← $dpyarea) + dacnt * dal                    01839
       ELSE                                                         01840
          daend ← &da + dal;                                        01841
       DO                                                           01842
          IF da.daexis AND da.dahandle AND NOT da.daseq AND NOT
          da.daauxiliary THEN                                       01843
             BEGIN                                                  01844
             &index ← da.dalsrt - lsrtl;                            01845
             end ← da.dalsz * lsrtl + da.dalsrt;                    01846
             %free storage associated with strings%                 01847
                UNTIL (&index ← &index + lsrtl) >= end OR NOT
                index.rtexis DO                                     01848
                   BEGIN                                            01849
                   IF free AND (index.rtsrce = srcstno OR index.rtsrce =
                   srcsig) THEN                                     01850
                      BEGIN                                         01851
                      straddr ← index.rtbps.bpadr;                  01852
                      <STGMGT, freestring> (straddr - 1, $dspblk);  01853
                      END;                                          01854
                   %zero entries - very important%                  01855
                      FOR count ← 0 UP  UNTIL =  lsrtl DO           01856
                      (&index + count) ← 0;                         01857
                      END;                                          01858
             used ← &index - da.dalsrt;                             01859
             lines ← da.damrow/da.davinc + 2;%number of useable lines +
```

```
            1%                                                          01860
            IF (oldsize ← da.dalsz ) > lines AND used/lsrtl < lines
            THEN                                                        01861
               BEGIN                                                    01862
               IF (da.dashrinkcnt ← da.dashrinkcnt + 1) > shrink THEN
                                                                        01863
                  BEGIN                                                 01864
                  oldtable ← (da.dalsrt := 0);                          01865
                  IF NOT maklsrt($dspblk, lines, &da) %no room in zone%
                  OR NOT oldtable %system error% THEN                   01866
                     BEGIN                                              01867
                     dismes(2, $"Allocator out of display space");      01868
                     da.dalsrt ← oldtable; %restore%                    01869
                     END                                                01870
                  ELSE                                                  01871
                     BEGIN                                              01872
                     mvbfbf(oldtable, da.dalsrt, oldsize * lsrtl);      01873
                     freeblk(oldtable - bhl, $dspblk);                  01874
                     oldtable ← aulsrt;                                 01875
                     IF NOT (aulsrt ← getblk(lines * lsrtl, $dspblk))
                     THEN                                               01876
                        BEGIN                                           01877
                        aulsrt ← oldtable;%use old table for aux%       01878
                        dismes(2, $"Non-fatal Display error due to
                        space
                        considerations.  Try viewspec 'f'.");           01879
                        END                                             01880
                     ELSE                                               01881
                        BEGIN                                           01882
                        freeblk(oldtable - bhl, $dspblk);               01883
                        aulsrt ← aulsrt + bhl; %start of useable space%
                                                                        01884
                        END;                                            01885
                     END;                                               01886
                  END                                                   01887
               ELSE da.dashrinkcnt ← 0;                                 01888
               END;                                                     01889
            da.dacrow ← da.daccol ← 0;                                  01890
            END                                                         01891
      UNTIL (&da ← &da + dal) >= daend;                                 01892
      RETURN;                                                           01893
      END.                                                              01894
                                                                        01895
%...special da functions...%                                           01896
   (clearda) PROCEDURE(da); %clear display area%                        05195
      %If &da is zero then zap the display image of all of the text
      display areas.  Otherwise, zap only the given text display area
      image.%                                                           05196
      %-----------------%                                              05197
      LOCAL  end, line, ls, dls, all, dacount, tempda, length;          05198
      LOCAL STRING send[100];                                           05199
      REF da, ls, dls, tempda, ltvsda;                                  05200
      IF NOT &da THEN                                                   05201
         BEGIN                                                          05202
         end ← (&da ← $dpyarea) + dacnt*dal;                            05203
         all ← TRUE;                                                    05204
```

```
    END                                                                 05205
ELSE                                                                    05206
    BEGIN                                                               05207
    end ← &da + dal;                                                    05208
    all ← FALSE;                                                        05209
    dacount ← 0;                                                        05210
    FOR &tempda ← $dpyarea UP dal UNTIL >= $dpyarea + dacnt*dal DO
                                                                        05211
       IF tempda.daexis AND NOT tempda.daauxiliary THEN BUMP
       dacount;                                                         05212
    END;                                                                05213
CASE nldevice OF                                                        05214
    = devlproc:                                                         05215
       BEGIN                                                            05216
       IF all OR dacount = 1 THEN %clear the screen and rewrite
       command lines%                                                   05217
          BEGIN                                                         05218
          % check first if there is anything in text area to
          erase%                                                        05219
             IF da.dacrow = 0 AND da.daccol = 0 THEN RETURN;            05220
          cscreen();                                                    05221
          IF (lptype .A %M) = deltadata THEN                            05222
             cline(0, ltvsda.datop, lpxmax);                            05223
          msgreset ← TRUE;                                              05224
          *send* ← *dnstr*;                                             05225
          dn($send);                                                    05226
          dsubsys($ssysname);                                           05227
          *savcfl* ← NULL;                                              05228
          cfldsp();                                                     05229
          IF arowon THEN                                                05230
             BEGIN                                                      05231
             arowon ← FALSE;                                            05232
             an();                                                      05233
             END                                                        05234
          ELSE                                                          05235
             BEGIN                                                      05236
             arowon ← TRUE;                                             05237
             af();                                                      05238
             END;                                                       05239
          vspsav ← vspsav[1] ← 0;                                       05240
          dspvsp(da.davspec, da.davspc2, 3);                            05241
          &ls ← da.dalsrt;                                              05242
          UNTIL NOT ls.rtexis DO                                        05243
             BEGIN                                                      05244
             ls.rtlsid ← 0;                                             05245
             &ls ← &ls + lsrtl;                                         05246
             END;                                                       05247
          DO da.dahandle ← 1                                            05248
          UNTIL (&da ← &da+dal) >= end;                                 05249
          END                                                           05250
       ELSE                                                             05251
          DO                                                            05252
             IF da.daexis AND da.dahandle THEN                          05253
                BEGIN %delete all of the lines%                         05254
                &ls ← da.dalsT;                                        05255
                UNTIL NOT ls.rtexis DO                                  05256
```

```
                    BEGIN                                                 05257
                    line ← ls.rty;                                        05258
                    &dls ← &ls;                                           05259
                    LOOP                                                  05260
                       BEGIN                                              05261
                       dls.rtlsid ← 0;                                    05262
                       IF dls.rtlplink THEN &dls ← &dls +
                       dls.rtlplink                                       05263
                       ELSE EXIT LOOP;                                    05264
                       END;                                               05265
                    IF (length ← (dls.rtx2-ls.rtxl)/da.dahinc + 1)
                    IN /1, lpxmax/ THEN                                   05266
                       cline(da.daleft+ls.rtxl, da.datop + line,
                       length);                                           05267
                    &ls ← &dls + lsrtl;                                   05268
                    END;                                                  05269
                 da.dahandle ← 1;                                         05270
                 END .                                                    05271
           UNTIL (&da ← &da+dal) >= end;                                  05272
        END;                                                              05273
= imlac0, =imlac1:                                                        05274
        BEGIN                                                             05275
        DO                                                                05276
           IF da.daexis AND da.dahandle THEN                              05277
              BEGIN                                                       05278
              % lst suppress display area%                                05279
                 *send* ← begmsg, 4+remfudge, remsdda,
                 da.dahandle.daidr1 + remfudge,
                 da.dahandle.daidr2 + remfudge, 1;                        05280
                 !sout(dspjfn, chbmty + $send ,-send.L);                  05281
              %then  restore it%                                          05282
                 *send* ← begmsg, 3+remfudge, remrdda,
                 da.dahandle.daidr1 + remfudge,
                 da.dahandle.daidr2 + remfudge;                           05283
                 !sout(dspjfn, chbmty + $send ,-send.L);                  05284
              IF ldspjfn THEN                                             05285
                 BEGIN                                                    05286
                 %suppress linked guy too%                                05287
                    *send* ← begmsg, 4+remfudge, remsdda,
                    da.dalhandle.daidr1 + remfudge,
                    da.dalhandle.daidr2 + remfudge, 1;                    05288
                    !sout(ldspjfn, chbmty + $send ,-send.L);              05289
                 %then  restore it%                                       05290
                    *send* ← begmsg, 3+remfudge, remrdda,
                    da.dalhandle.daidr1 + remfudge,
                    da.dalhandle.daidr2 + remfudge;                       05291
                    !sout(ldspjfn, chbmty + $send ,-send.L);              05292
                 END;                                                     05293
              %clear lsid bit table%                                      05294
                 dassnbit(da.dalsidb, -1, lsbtsize);                      05295
                    %bit num of -1 means clear all bits%                  05296
              END                                                         05297
        UNTIL (&da ← &da+dal) >= end;                                     05298
        END;                                                              05299
     ENDCASE                                                              05300
        BEGIN                                                             05301
```

```
        DO                                                              05302
           IF da.daexis AND da.dahandle THEN                            05303
              BEGIN                                                     05304
              r1 ← da.dahandle;                                         05305
              r2 ← 1;                                                   05306
              IF NOT SKIP !JSYS sdda THEN                               05307
                 err($"SDDA jsys error, clearda");                      05308
              IF NOT SKIP !JSYS rdda THEN                               05309
                 err($"RDDA jsys error, clearda");                      05310
              IF da.dalhandle THEN                                      05311
                 BEGIN                                                  05312
                 r1 ← da.dalhandle;                                     05313
                 r2 ← 1;                                                05314
                 IF NOT SKIP !JSYS sdda THEN                            05315
                    err($"Link SDDA jsys error, clearda");              05316
                 IF NOT SKIP !JSYS rdda THEN                            05317
                    err($"Link RDDA jsys error, clearda");              05318
                 END;                                                   05319
              END                                                       05320
        UNTIL (&da ← &da+dal) >= end;                                   05321
        END;                                                            05322
   RETURN;                                                              05323
   END.

                                                                        05324
(supda) PROCEDURE(da); %suppress display area%                          02024
   %If &da is zero then suppress the display image of all of the
   text display areas.  Otherwise, suppress only the given text
   display area image.%                                                 02025
   %----------------%                                                  02026
   LOCAL  end, line, ls, dls, all, dacount, tempda, length;             02027
   LOCAL STRING send[200];                                              02028
   REF da, ls, dls, tempda, ltvsda;                                     02029
   IF NOT &da THEN                                                      02030
      BEGIN                                                             02031
      end ← (&da ← $dpyarea) + dacnt*dal;                               02032
      all ← TRUE;                                                       02033
      END                                                               02034
   ELSE                                                                 02035
      BEGIN                                                             02036
      end ← &da + dal;                                                  02037
      all ← FALSE;                                                      02038
      dacount ← 0;                                                      02039
      FOR &tempda ← $dpyarea UP dal UNTIL >= $dpyarea + dacnt*dal DO
                                                                        02040
         IF tempda.daexis THEN BUMP dacount;                            02041
      END;                                                              02042
   CASE nldevice OF                                                     02043
      = devlproc:                                                       02044
         BEGIN                                                          02045
         IF all OR dacount = 1 THEN %clear the screen and rewrite
         command lines for single da %                                  02046
            BEGIN                                                       02047
            cscreen();                                                  02048
            IF (lptype .A 4M) = deltadata THEN                          03367
               cline(0, ltvsda.datop, lpxmax);                          03368
            dn($dnstr);                                                 02049
```

```
        *savcfl* ← NULL;                                                02050
        cfldsp();                                                       02051
        IF arowon THEN                                                  02052
           BEGIN                                                        02053
           arowon ← FALSE;                                              02054
           an();                                                        02055
           END                                                          02056
        ELSE                                                            02057
           BEGIN                                                        02058
           arowon ← TRUE;                                               02059
           af();                                                        02060
           END;                                                         02061
        vspsav ← vspsav[1] ← 0;                                         02062
        dspvsp(da.davspec, da.davspc2, 3);                              02063
        &ls ← da.dalsrt;                                                02064
        UNTIL NOT ls.rtexis DO                                          02065
           BEGIN                                                        02066
           ls.rtlsid ← 0;                                               02067
           &ls ← &ls + lsrtl;                                           02068
           END;                                                         02069
        DO                                                              02070
           BEGIN                                                        02071
           da.dahandle ← 1;                                             02072
           da.dasuppress ← TRUE;                                        02073
           END                                                          02074
        UNTIL (&da ← &da+dal) >= end;                                   02075
        END                                                             02076
     ELSE                                                               02077
        DO                                                              02078
           IF da.daexis AND da.dahandle THEN                            02079
              BEGIN %delete all of the lines%                           02080
              &ls ← da.dalsrt;                                          02081
              UNTIL NOT ls.rtexis DO                                    02082
                 BEGIN                                                  02083
                 line ← ls.rty;                                         02084
                 &dls ← &ls;                                            02085
                 LOOP                                                   02086
                    BEGIN                                               02087
                    dls.rtlsid ← 0;                                     02088
                    IF dls.rtlplink THEN &dls ← &dls +
                    dls.rtlplink                                        02089
                    ELSE EXIT LOOP;                                     02090
                    END;                                                02091
                 IF (length ← (dls.rtx2-ls.rtxl)/da.dahinc + 1)
                 IN [1, lpxmax] THEN                                    02696
                    cline(da.daleft+ls.rtxl, da.datop + line,
                    length);                                            02092
                 &ls ← &dls + lsrtl;                                    02094
                 END;                                                   02095
              da.dahandle ← 1;                                          02096
              da.dasuppress ← TRUE;                                     02097
              END                                                       02098
        UNTIL (&da ← &da+dal) >= end;                                   02099
     END;                                                               02100
= im1ac0, = im1ac1:                                                     02101
     BEGIN                                                              02102
```

```
        DO                                                              02103
           IF da.daexis AND NOT da.daseq AND da.dahandle AND NOT
           da.daauxiliary THEN                                          02104
              BEGIN                                                     02105
              da.dasuppress ← TRUE;                                     02106
              *send* ← begmsg, 4+remfudge, remsdda,                     02107
                 da.dahandle.daidr1 + remfudge,                         02108
                 da.dahandle.daidr2 + remfudge,                         02109
                 FALSE;                                                 02110
              !sout(dspjfn, chbmty + $send ,-send.L);                   02111
              IF da.dalhandle THEN                                      02112
                 BEGIN                                                  02113
                 %suppress linked guy too%                              02114
                    *send* ← begmsg, 4+remfudge, remsdda,               02115
                       da.dalhandle.daidr1 + remfudge,                  02116
                       da.dalhandle.daidr2 + remfudge,                  02117
                       FALSE;                                           02118
                    !sout(ldspjfn, chbmty + $send ,-send.L);            02119
                 END;                                                   02120
              END                                                       02121
        UNTIL (&da ← &da+dal) >= end;                                   02122
        END;                                                            02123
     ENDCASE                                                            02124
        BEGIN                                                           02125
        DO                                                              02126
           IF da.daexis AND NOT da.daseq AND da.dahandle AND NOT
           da.daauxiliary THEN                                          02127
              BEGIN                                                     02128
              da.dasuppress ← TRUE;                                     02129
              r1 ← da.dahandle;                                         02130
              r2 ← 0;                                                   02131
              IF NOT SKIP !JSYS sdda THEN                               02132
                 err($"sdda jsys error, supda");                        02133
              IF da.dalhandle THEN                                      02134
                 BEGIN                                                  02135
                 r1 ← da.dalhandle;                                     02136
                 r2 ← 0;                                                02137
                 IF NOT SKIP !JSYS sdda THEN                            02138
                    err($"Link sdda jsys error, supda");                02139
                 END;                                                   02140
              END                                                       02141
        UNTIL (&da ← &da+dal) >= end;                                   02142
        END;                                                            02143
  RETURN;                                                               02144
  END.

                                                                        02145
(resda) PROCEDURE(da); %restore display area%                           02146
  %If &da is zero then restore the display image of all of the text
  display areas.  Otherwise, restore only the given text display
  area image.%                                                          02147
  %----------------%                                                   02148
  LOCAL  end;                                                           02149
  LOCAL STRING send[200];                                               02150
  REF da;                                                               02151
  IF NOT &da THEN                                                       02152
     end ← (&da ← $dpyarea) + dacnt*dal                                 02153
```

```
ELSE                                                                    02154
   end ← &da + dal;                                                     02155
CASE nldevice OF                                                        02156
   = devlproc:                                                          02157
      BEGIN                                                             02158
      DO                                                                02159
         IF da.daexis AND NOT da.daseq AND da.dahandle AND
         da.dasuppress THEN                                             02160
            BEGIN                                                       02161
            da.dasuppress ← FALSE;                                      02162
            %call the formatter%                                        02163
               dafrmt(&da, 0);                                          02164
            END                                                         02165
      UNTIL (&da ← &da+dal) >= end;                                     02166
      END;                                                              02167
   = imlac0, = imlac1:                                                  02168
      BEGIN                                                             02169
      DO                                                                02170
         IF da.daexis AND NOT da.daseq AND da.dahandle AND
         da.dasuppress AND NOT da.daauxiliary THEN                      02171
            BEGIN                                                       02172
            %send command to restore it%                                02173
               *send* ← begmsg, 3+remfudge, remrdda,
               da.dahandle.daidr1 + remfudge,
               da.dahandle.daidr2 + remfudge;                           02174
               !sout(dspjfn, chbmty + $send ,-send.L);                  02175
            IF da.dalhandle THEN                                        02176
               BEGIN                                                    02177
               %restore linked guy too%                                 02178
                  *send* ← begmsg, 3+remfudge, remrdda,
                  da.dalhandle.daidr1 + remfudge,
                  da.dalhandle.daidr2 + remfudge;                       02179
                  !sout(idspjfn, chbmty + $send ,-send.L);              02180
               END;                                                     02181
            da.dasuppress ← FALSE;                                      02182
            END                                                         02183
      UNTIL (&da ← &da+dal) >= end;                                     02184
      END;                                                              02185
   ENDCASE                                                              02186
      BEGIN                                                             02187
      DO                                                                02188
         IF da.daexis AND NOT da.daseq AND da.dahandle AND
         da.dasuppress AND NOT da.daauxiliary THEN                      02189
            BEGIN                                                       02190
            r1 ← da.dahandle;                                           02191
            IF NOT SKIP !JSYS rdda THEN                                 02192
               err($"rdda jsys error, resda");                          02193
            IF da.dalhandle THEN                                        02194
               BEGIN                                                    02195
               r1 ← da.dalhandle;                                       02196
               IF NOT SKIP !JSYS rdda THEN                              02197
                  err($"Link rdda jsys error, resda");                  02198
               END;                                                     02199
            da.dasuppress ← FALSE;                                      02200
            END                                                         02201
      UNTIL (&da ← &da+dal) >= end;                                     02202
```

```
        END;                                                        02203
    RETURN;                                                         02204
    END.
                                                                    02205
(alocda) PROCEDURE (da); %allocate a display area%                  02206
    %uses the information in the display area record to allocate the
    display area.  The system display area handle is stored in the
    dahandle field.%                                                02207
    %-----------------%                                            02208
    LOCAL save;                                                     02209
    LOCAL STRING send[20];                                          02210
    REF da;                                                         02211
    CASE nldevice OF                                                02212
        = devlproc:                                                 02213
            BEGIN                                                   02214
            da.dahandle ← 1;                                        02215
            da.dalhandle ← 0;                                       02216
            END;                                                    02217
        = imlac0, = imlac1:                                         02218
            BEGIN                                                   02219
            da.dahandle ← assnbit(@dabt, dabtsize);                 02220
            IF da.dalsidb THEN                                      02221
                dassnbit(da.dalsidb, -1, lsbtsize);                 02222
            *send* ← begmsg, 14+remfudge, nwada,                    02223
                %daid%                                              02224
                    da.dahandle.daidr1 + remfudge,                  02225
                    da.dahandle.daidr2 + remfudge,                  02226
                %number of lines%                                   02227
                    (da.dabottom - da.datop)/da.davinc,             02228
                %number of columns%                                 02229
                    (da.daright - da.daleft)/da.dahinc,             02230
                %coords of lower left corner -- y must be reversed here
                (in NLS 0,0 is UPPER left, for display 0,0 is LOWER
                left.  Also must divide by 256 for display resolution.)%
                                                                    02231
                    (da.daleft/256).xc1 + remfudge,                 02232
                    (da.daleft/256).xc2 + remfudge,                 02233
                    (save ←(bmarg-da.dabottom)/256).yc1 + remfudge, 02234
                    save.yc2 + remfudge,                            02235
                %vinc%                                              02236
                    (da.davinc/256).xc1 + remfudge,                 02237
                    (da.davinc/256).xc2 + remfudge,                 02238
                %format defaults%                                   02239
                    da.dacsize, da.dahinc/256, da.dafont;           02240
            !sout(dspjfn, chbnty+@send, -send.L);                   02241
            END;                                                    02242
        ENDCASE                                                     02243
            BEGIN                                                   02244
            r1.adauly ← (bmarg-tmarg-da.datop)/256;                 02245
            r1.adaulx ← da.daleft/256;                              02246
            r1.adasize ← da.dalsz;                                  02247
            r1.adaseq ← da.daseq;                                   02248
            save ← r1;                                              02249
            r2.adalry ← (bmarg-tmarg-da.dabottom)/256;              02250
            r2.adalrx ← da.daright/256;                             02251
            r2.adadcs ← da.dacsize;                                 02252
```

```
            r2.adadhi ← da.dahinc/256;                                  02253
            r2.adadf ← da.dafont;                                       02254
            da.dahandle ← da.dalhandle ← 0;                             02255
            IF NOT SKIP !JSYS ada THEN                                  02256
               IF r1 = syserr THEN NULL                                 02257
               ELSE dismes(2, $"ADA jsys error")                        02258
            ELSE da.dahandle ← r1 .A 18M;                               02259
            IF linkcnsl THEN %terminal display link--allocate another
            da%                                                         02260
               BEGIN                                                    02261
               r1 ← save;                                               02262
               r3 ← linkcnsl;                                           02263
               IF NOT SKIP !JSYS nada THEN                              02264
                  IF r1 = syserr THEN NULL                              02265
                  ELSE dismes(2, $"Link NADA jsys error")               02266
               ELSE da.dalhandle ← r1 .A 18M;                           02267
               END;                                                     02268
            END;                                                        02269
      RETURN(da.dahandle);                                              02270
      END.
                                                                        02271
   (dealocda) PROCEDURE (da);  %given a system daid, deallocate it%
                                                                        02272
      LOCAL STRING send[20];                                            02273
      REF da;                                                           02274
      CASE nldevice OF                                                  02275
         = devlproc: NULL;                                              02276
         = imlac0, = imlac1: %send out string%                          02277
            BEGIN                                                       02278
            *send* ← begmsg, 3+remfudge, remdda,                        02279
               da.dahandle.daidr1+remfudge,                             02280
               da.dahandle.daidr2+remfudge;                             02281
            !sout(dspjfn, chbmty+$send, -send.L);                       02282
            dassnbit($dabt, da.dahandle, dabtsize);                     02283
            da.dahandle ← 0;                                            02284
            END;                                                        02285
         ENDCASE %local tasker%                                         02286
            BEGIN                                                       02287
            IF da.dahandle THEN                                         02288
               IF NOT SKIP !dda(da.dahandle := 0) THEN err($"DDA JSYS
               failed, DEALOCDA");                                      02289
            IF da.dalhandle THEN                                        02290
               IF NOT SKIP !ndda(da.dalhandle := 0, linkcnsl) THEN
               err($"NDDA JSYS failed, DEALOCDA");                      02291
            END;                                                        02292
      RETURN;                                                           02293
      END.                                                              02294
                                                                        02295
%...graphics primitives for alphanumeric (line-processor) displays %
                                                                        02296
   (lpdspstr) PROCEDURE (x, y, string, standoutflag);  %display the
   specified string at specified location on a line processor
   alpha-numeric display.  IF x = current then use current position of
   cursor.  If standoutflag is TRUE then cause the string to standout%
                                                                        02297
      LOCAL store, start, char, l, mx;                                  02298
```

```
    LOCAL STRING send[200];                                              02299
    REF string;                                                          02300
    IF x NOT= current THEN position(x, y);                               02301
    IF standoutflag THEN standout();                                     02302
    CCPOS SF(*string*);                                                  02303
    start ← store ← chbmty + $send;                                      02304
    *send* ← NULL;                                                       02305
    l ← send.L;  mx ← send.M;                                            02306
    LOOP                                                                 02307
       BEGIN                                                             02308
       CASE char ← READC OF                                              02309
          = ENDCHR: EXIT LOOP;                                           02310
          = TAB:                                                         02311
             BEGIN                                                       02312
             ↑store ← lptab; BUMP l;  %*send* ← *send*, lptab;%          02313
             END;                                                        02314
          ENDCASE                                                        02342
             BEGIN                                                       02343
             ↑store ← char; BUMP l; %*send* ← *send*, char;%             02344
             END;                                                        02345
       IF l >= mx THEN %time to sout%                                    02346
          BEGIN                                                          02347
          !sout(dspjfn, start, -1);                                      02348
          IF ldspjfn THEN                                                03314
             !sout(ldspjfn, start, -1);                                  03315
          *send* ← NULL;                                                 02349
          l ← send.L;                                                    02350
          store ← start;                                                 02351
          END;                                                           02352
       END;                                                              02353
    IF l THEN %output it%                                                02354
       BEGIN                                                             03317
       !sout(dspjfn, start, -1);                                         02355
       IF ldspjfn THEN                                                   03316
          !sout(ldspjfn, start, -1);                                     03319
       END;                                                              03318
    IF standoutflag THEN endstandout();                                  02356
    track();                                                             02357
    RETURN;                                                              02358
    END.                                                                 02359

                                                                         02360
(pad) PROCEDURE (count);  %pad with count null characters%               02361
    %adjusts for current line speed by dividing count by
    lpbaudfactor%                                                        02362
    count ← MIN((count/lpbaudfactor) + 1, padstr.M);                     02363
    IF count > 0 THEN                                                    02364
       BEGIN                                                             03320
       !sout(dspjfn, chbmty + $padstr, -count);                          02365
       IF ldspjfn THEN                                                   03322
          !sout(ldspjfn, chbmty + $padstr, -count);                      03323
       END;                                                              03321
    RETURN;                                                              02366
    END.                                                                 02367

                                                                         02368
```

```
(position) PROCEDURE % position cursor - alphanumeric displays %
                                                                    02369
   ( xpos,  % x position %                                          02370
     ypos); % y position %                                          02371
   LOCAL STRING send[10];  %for sout%                               02372
   IF xpos NOT IN [0, lpxmax] OR ypos NOT IN [0, lpymax] THEN       02691
      err($"Illegal coordinate detected in POSITION");              02692
   *send* ← lpesc, lpposition, xpos+40B, lpymax-ypos+40B;           02373
   tracking ← FALSE;                                                02375
   !sout(dspjfn, chbmty + $send, -send.L);                          02374
   IF ldspjfn THEN                                                  03324
      !sout(ldspjfn, chbmty + $send, -send.L);                      03325
   RETURN;                                                          02376
   END.                                                             02377

                                                                    02378
(lpttywindow) PROCEDURE % specify tty window - alphanumeric displays
%                                                                   02379
   ( top,   % top line of window %                                  02380
     bottom         % bottom line of window %                       02381
   );                                                               02382
   LOCAL save;                                                      02775
   LOCAL STRING send[10];                                           02383
   IF top NOT IN [0, lpymax] OR bottom NOT IN [0, lpymax] THEN      02697
      err($"Illegal coordinate detected in LPTTYWINDOW");           02698
   *send* ← lpesc, lptty, lpymax-top+40B, lpymax-bottom+40B;        02384
   save ← tracking := FALSE; %to avoid printer output%              02773
   !sout(dspjfn, chbmty + $send, -send.L);                          02385
   IF ldspjfn THEN                                                  03326
      !sout(ldspjfn, chbmty + $send, -send.L);                      03327
   tracking ← save; %to avoid printer output%                       02774
   RETURN;                                                          02386
   END.                                                             02387

                                                                    02388
(lprset) PROCEDURE; % reset alphanumeric displays %                 02389
   LOCAL save;                                                      02778
   LOCAL STRING send[10];                                           02390
   *send* ← lpesc, lpreset;                                         02391
   save ← tracking := FALSE; %to avoid printer output%              02776
   !sout (dspjfn, chbmty + $send, -send.L);                         02392
   IF ldspjfn THEN                                                  03328
      !sout (ldspjfn, chbmty + $send, -send.L);                     03329
   pad(lpdlpad);                                                    02393
   cscreen(); % be sure screen cleared %                            02394
   tracking ← save; %to avoid printer output%                       02777
   RETURN;                                                          02395
   END.                                                             02396

                                                                    02397
(cscreen) PROCEDURE; % clear screen - alphanumeric displays %       02398
   LOCAL save;                                                      02781
   LOCAL STRING send[10];                                           02399
   *send* ← lpesc, lpcscreen;                                       02400
   save ← tracking := FALSE; %to avoid printer output%              02779
   !sout(dspjfn, chbmty + $send, -send.L);                          02401
```

```
    IF ldspjfn THEN                                                    03330
       !sout(ldspjfn, chbmty + $send, -send.L);                        03331
    pad(lpdlpad);                                                      02402
    tracking ← save; %to avoid printer output%                         02780
    RETURN;                                                            02403
    END.                                                               02404

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

                                                                       02413
(cline) PROCEDURE % write blanks - alphanumeric displays %             02414
    (x,         %x coordinate or CURRENT%                              02415
    y,          %y coordinate or CURRENT%                              02416
    nchar );  % number of blanks to write %                            02417
    LOCAL STRING send[10];                                             02418
    IF nchar NOT IN [1,lpxmax] THEN err($"Illegal number of blanks
    requested in CLINE");                                              02690
    IF x NOT= current THEN position(x, y);                             02419
    *send* ← lpesc, lpcline, nchar+40B;                                02420
    !sout(dspjfn, chbmty + $send, -send.L);                            02421
    IF ldspjfn THEN                                                    03334
       !sout(ldspjfn, chbmty + $send, -send.L);                        03335
    pad(nchar);                                                        02422
    IF x NOT= current THEN track();                                    02423
    RETURN;                                                            02424
    END.                                                               02425

                                                                       02426
(dline) PROCEDURE % delete line - alphanumeric displays %              02427
    (x,       %x coordinate or CURRENT%                                02428
    y);       %y coordinate or CURRENT%                                02429
    LOCAL STRING send[10];                                             02430
    IF x NOT= current THEN position(x, y);                             02431
    *send* ← lpesc, lpdline;                                           02432
    !sout(dspjfn, chbmty + $send, -send.L);                            02433
    IF ldspjfn THEN                                                    03336
       !sout(ldspjfn, chbmty + $send, -send.L);                        03337
    pad(lpdlpad);                                                      02434
    track();                                                           02435
    RETURN;                                                            02436
    END.                                                               02437

                                                                       02438
(inline) PROCEDURE % insert line - alphanumeric displays %             02439
    (x,               %x coordinate or CURRENT%                        02440
    y.                %y coordinate or CURRENT%                        02441
```

```
    string,          %string to be displayed%                          02442
    standoutflag); %FLAG -- TRUE: make the string standout%            02443
    LOCAL STRING send[10];                                             02444
    REF string;                                                        02445
    IF x NOT= current THEN position(x, y);                             02446
    *send* ← lpesc, lpinline;                                          02447
    !sout(dspjfn, chbmty + $send, -send.L);                            02448
    IF ldspjfn THEN                                                    03339
       !sout(ldspjfn, chbmty + $send, -send.L);                        03340
    %pad(lpilpad);%                                                    03338
    lpdspstr( current, current, &string, standoutflag);                02449
    pad(30);                                                           03369
    RETURN;                                                            02450
    END.                                                               02451

                                                                       02452
(lpmarkit) PROCEDURE % bug selection mark - alphanumeric displays %
                                                                       02453
    ( xpos,  % x position of char to bug %                             02454
      ypos); % y position of char to bug %                             02455
    LOCAL save;                                                        02784
    LOCAL STRING send[10];                                             02457
    IF xpos NOT IN [0, lpxmax] OR ypos NOT IN [0, lpymax] THEN         02700
       err($"Illegal coordinate detected in POSITION");                02701
    *send* ← lpesc, lpbug, xpos+40B, lpymax-ypos+40B;                  02458
    save ← tracking := FALSE; %to avoid printer output%                02782
    !sout(dspjfn, chbmty + $send, -send.L);                            02459
    IF ldspjfn THEN                                                    03341
       !sout(ldspjfn, chbmty + $send, -send.L);                        03342
    pad(8); %change to use a symbolic value!!!%                        02460
    tracking ← save; %to avoid printer output%                         02783
    RETURN;                                                            02461
    END.                                                               02462

                                                                       02463
(lppopmark) PROCEDURE; % pop bug selection mark - alphanumeric
displays %                                                             02464
    LOCAL save;                                                        02785
    LOCAL STRING send[10];                                             02465
    *send* ← lpesc, lppbug;                                            02466
    save ← tracking := FALSE; %to avoid printer output%                02786
    !sout(dspjfn, chbmty + $send, -send.L);                            02467
    IF ldspjfn THEN                                                    03343
       !sout(ldspjfn, chbmty + $send, -send.L);                        03344
    pad(8); %change to use a symbolic value!!!%                        02468
    tracking ← save; %to avoid printer output%                         02787
    RETURN;                                                            02469
    END.                                                               02470

                                                                       02471
(intter) PROCEDURE; % interrogate alphanumeric displays %              02472
    LOCAL save;                                                        02790
    LOCAL STRING send[10];                                             02473
    *send* ← lpesc, lpintter;                                          02474
    save ← tracking := FALSE; %to avoid printer output%                02789
    !sout(dspjfn, chbmty + $send, -send.L);                            02475
```

```
   IF ldspjfn THEN                                                    03345
      !sout(ldspjfn, chbmty + $send, -send.L);                        03346
   tracking ← save; %to avoid printer output%                         02788
   RETURN;                                                            02476
   END.                                                               02477

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

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

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

                                                                      02499
(lppopen) PROCEDURE; % open Lineprocessor printer %                   02535
   LOCAL STRING send[10];                                             02536
   *send* ← lpesc, lptptopen;                                         02537
   !sout(dspjfn, chbmty + $send, -send.L);                            02538
   %dont do shared screen sout!%                                      03357
   RETURN;                                                            02539
   END.                                                               02540

                                                                      02541
(lppclose) PROCEDURE; % close Lineprocessor printer %                 02542
   LOCAL STRING send[10];                                             02543
   *send* ← lpesc, lptptclose;                                        02544
   !sout(dspjfn, chbmty + $send, -send.L);                            02545
   RETURN;                                                            02546
```

```
   END.                                                                 02547

                                                                        02548
(lppsr) PROCEDURE % Lineprocessor printer string handler %              02549
   (lppcnt); % number of characters to send %                           02550
   % sends a string of lppcnt chars to Line Processor printer.  Gets
   them from procedure lppch %                                          02551
   LOCAL STRING send[200];                                              02552
   LOCAL store, l;                                                      02553
   IF &lppch=0 THEN err($"no LP character routine - lppsr");            02554
   IF lppcnt<=0 OR lppjfn=0 THEN RETURN;                                02555
   *send* ← NULL;                                                       02556
   l ← send.L;                                                          02557
   store ← chbmty + $send;                                              02558
   ↑store ← lpesc;                                                     02559
   ↑store ← lptptsr;                                                   02560
   ↑store ← 40B;                                                       02561
   ↑store ← lppcnt+40B;                                                02562
   l ← l + 4;                                                           02563
   WHILE (lppcnt ← lppcnt-1) >= 0 DO                                    02564
      BEGIN                                                             02565
      ↑store ← lppch(); % get character and store it %                  02566
      BUMP l;                                                           02567
      END;                                                              02568
   !sout(dspjfn, chbmty+$send, l);                                      02569
   !gtsts(lppjfn);                                                      02570
   IF r2 .A 1B9 THEN % end of file! %                                   02571
      lppterm();                                                        02572
   RETURN;                                                              02573
   END.                                                                 02574
(lppterm) PROCEDURE; % terminate LP printing %                          02575
   % aborts Line Processor printer operation %                         02576
   !dti(0); % deassign null %                                           02746
   !dic(4B5,200B); % deactivate chan 28 %                               02748
   lppclose();                                                          02577
   IF NOT SKIP !closf(lppjfn) THEN err($"cant close text file");
                                                                        02578
   lppjfn ← tracksend ← 0;                                              02579
   RETURN;                                                              02580
   END.                                                                 02581
                                                                        02744
(lppstart) PROCEDURE % start LP printing from jfn %                     02729
   (jfn, % jfn for text file %                                          02730
   proc); % address of procedure to get chars from %                    02731
   % call this procedure to start printing on LP printer %             02732
   % setup variables %                                                 02737
      lppjfn ← jfn;                                                     02733
      &lppch ← proc;                                                    02734
      tracksend ← lppcol ← lpplin ← 0;                                  02735
   % setup PSI stuff for string requests from LP %                     02736
      chntab[28] ← 1B6+$lppint; % use channel 28 for interrupt %
                                                                        02738
      !ati(0+28); % designate null (zero) as interrupt char %           02739
      !aic(4B5,200B); % activate chn 28 %                               02740
   lppopen(); % send open string %                                      02741
   RETURN;                                                              02742
```

```
   END.                                                                  02743
                                                                         02745
(lppint) PROCEDURE; % LP printer interupt routine%                       02751
   %---------------------%                                               02752
                                                                         02753
   %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%                    02754
                                                                         02755
   %---------------------%                                               02756
   %save registers%                                                      02757
   svac1 ← r1;                                                           02758
   r1 ← $svacs;                                                          02759
   !BLT r1, svacse;                                                      02760
   !MOVE r1,40B; sav40 ← r1;                                             02769
   s ← -100; !HRL s,s; !HRRI s,psistk;                                   02771
   m ← s;                                                                02772
   IF tracking THEN lppsr(16)                                            02761
   ELSE tracksend ← tracksend + 16;                                      02762
   %restore registers%                                                   02763
   r1 ← sav40; !MOVEM r1,40B;                                            02770
   !HRLZI r1, svacs;                                                     02764
   !BLT r1, 17B;                                                         02765
   r1 ← svac1;                                                           02766
   !debrk;                                                               02767
   END.                                                                  02768
FINISH of DISPLAY                                                        02500
(wrtstr) PROCEDURE (da, ls); %write string in display area%              03116
   %Write a string onto the display.  Uses the information in the
   display area and line segment records to determiine what to write
   and where and how.  If the rtbps field is -1 then the old string is
   used.%                                                                03117
   %---------------------%                                               03118
   LOCAL nls, store, savestart, save, startbp, endbp, i, char, l, mx,
   blanks, slength, bp, length, format;                                  03119
   LOCAL STRING send[200];                                               03120
   REF da, ls, nls;                                                      03121
   CASE nldevice OF                                                      03122
      = devlproc:                                                        03123
         BEGIN                                                           03124
         %send position cursor code%                                     03125
            position( i ← ls.rtx1+da.daleft, ls.rty+da.datop );          03126
         %send text%                                                     03127
            LOOP                                                         03128
               BEGIN                                                     03129
               startbp ← ls.rtbps;                                       03130
               endbp ← ls.rtbpe;                                         03131
               save ← MIN(da.daright, ls.rtx2+da.daleft);                03132
                  %rtx2 points to the last char of the string%           03133
               savestart ← store ← chbmty + $send;                       03134
               *send* ← NULL;                                            03135
               l ← send.L;  mx ← send.M;                                 03136
               blanks ← 0;                                               03137
               FOR i UP ls.rthinc UNTIL > save DO                        03138
                  BEGIN                                                  03139
                  IF startbp = endbp THEN %fill with blanks%             03140
                     BUMP blanks                                         03141
```

```
            ELSE                                                    03142
               BEGIN                                                03143
               CASE (char ← ↑startbp) OF                            03144
                  = TAB: char ← lptab;                              03145
                  = EOL: char ← CR;                                 03146
                  ENDCASE;                                          03147
               %*send* ← *send*, char;%                             03148
                  ↑store ← char;                                    03149
                  BUMP ?L;                                          03150
               END;                                                 03151
            IF l >= mx THEN %time to sout -- string full%           03152
               BEGIN                                                03153
               !sout(dspjfn, savestart, -1);                        03154
               store ← savestart;                                   03155
               *send* ← NULL;                                       03156
               l ← send.L;                                          03157
               END;                                                 03158
            END;                                                    03159
         IF l THEN %sout the string%                                03160
            BEGIN                                                   03161
            !sout(dspjfn, savestart, -1);                           03162
            store ← savestart;                                      03163
            *send* ← NULL;                                          03164
            l ← send.L;                                             03165
            END;                                                    03166
         IF blanks IN [1, lpxmax] THEN cline(current, current,
         blanks);                                                   03167
         %check to see if several line segments are to be
         writtten on this line%                                     03168
            IF ls.rtlplink THEN  %there is another line segment
            -- loop again%                                         03169
               BEGIN                                                03170
               &nls ← &ls + ls.rtlplink;                            03171
               blanks ← nls.rtx1 - ls.rtx2 -1;                      03172
               IF blanks IN [1, lpxmax] THEN cline(current,
               current, blanks);                                    03173
               &ls ← &nls;                                          03174
               REPEAT LOOP;                                         03175
               END                                                  03176
            ELSE EXIT LOOP;                                         03177
         END;                                                       03178
   track();                                                         03179
   END;                                                             03180
= imlac0, = imlac1:                                                 03181
   BEGIN                                                            03182
   CASE ls.rtbps OF                                                 03183
      = 0: %delete string%                                          03184
         BEGIN                                                      03185
         IF ls.rtlsid = 0 OR NOT chkbit(da.dalsidp, ls.TLSID,
         lsbtsize) THEN BEGIN                                       03186
            dismes(2, $"wrtstr: suppressing a non-existant
            string");                                               03187
            RETURN;                                                 03188
            END;                                                    03189
         *send* ← begmsg, 5+remfudge, remssda,
         da.dahandle.daidr1 + remfudge,
```

```
      da.dahandle.daidr2 + remfudge, ls.rtlsid, TRUE;
      !sout(dspjfn, chbmty + $send ,-send.L);
      dassnbit(da.dalsidb,ls.rtlsid,lsbtsize);
      RETURN;
      END;
   = -1: %use old string%
      IF da.daseq THEN RETURN
      ELSE slength ← -1;
   ENDCASE %new string%
      BEGIN %get string length%
      startbp ← ls.rtbps;
      endbp ← ls.rtbpe;
      slength ←slngth(startbp, endbp);
      END;
%setup format info and assign or check stid
if default values, use default constants %
format ← 0;
save ← (da.dabottom-da.datop-ls.rty)/256;
IF ls.rtlsid = 0 THEN  %new string%
   BEGIN
   ls.rtlsid ←  assnbit(da.dalsidb,lsbtsize);
   format.xyds ← FALSE;
   END
ELSE
   BEGIN
   IF NOT chkbit(da.dalsidb,ls.rtlsid,lsbtsize) THEN
      BEGIN
      dismes(2,$"wrtstr: lsbitable fouled - CHKBIT failure!");

      RETURN;
      END;
   format.xyds ← IF ls.rtxl OR save THEN FALSE ELSE TRUE;
   END;
IF ls.rtfont = da.dafont THEN format.fdd ← TRUE
ELSE format.fdd ← format.fds ← FALSE;
IF ls.rtcsize = da.dacsize THEN format.sdd ← TRUE
ELSE format.sdd ← format.sds ← FALSE;
IF ls.rthinc = da.dahinc THEN format.idd ← TRUE
ELSE format.idd ← format.ids ← FALSE;
%send command parameters plus string%
   length ← slength ←
      MIN((da.daright-ls= rtxl)/ls.rthinc, slength); %safety
      precaution%
   IF da.daseq THEN % if sequential then use APPEND command%

      BEGIN
      *send* ← begmsg, length+3+remfudge, apsda,
        da.dahandle.daidr1+remfudge,
      da.dahandle.daidr2+remfudge;
      bp ← chbptr(send.L) + $send;
      END
   ELSE
      BEGIN
      %compute message length%
      length ← MAX(0, lengtn) + 6 + remfudge;
         IF NOT format.xyds THEN length ← length + 4;
```

```
            IF NOT format.sds AND NOT format.sdd THEN BUMP
            length;                                                  03240
            IF NOT format.ids AND NOT format.idd THEN BUMP
            length;                                                  03241
            IF NOT format.fds AND NOT format.fdd THEN BUMP
            length;                                                  03242
         %string manipulation as per (JOURNAL, 11726,)%              03243
            *send* ← begmsg, length,  nwstrda,
             da.dahandle.daidr1+remfudge,
            da.dahandle.daidr2+remfudge, ls.rtlsid,
             slength + 1, format + remfudge;                         03244
            bp ← chbptr(send.L) + $send;                             03245
         %cordinates%                                                03246
            IF NOT format.xyds THEN                                  03247
               BEGIN                                                 03248
               ↑bp ← remfudge + (ls.rtx1/256).xc1;                   03249
               ↑bp ← remfudge + (ls.rtx1/256).xc2;                   03250
               ↑bp ← remfudge + save.yc1;                            03251
               ↑bp ← remfudge + save.yc2;                            03252
               send.L ← send.L + 4;                                  03253
               END;                                                  03254
         %character size%                                            03255
            IF NOT format.sds AND NOT format.sdd THEN                03256
               BEGIN                                                 03257
               ↑bp ← ls.rtcsize;                                     03258
               BUMP send.L;                                          03259
               END;                                                  03260
         %horizontal increment%                                      03261
            IF NOT format.ids AND NOT format.idd THEN                03262
               BEGIN                                                 03263
               ↑bp ← (ls.rthinc/128) + 1;                            03264
               BUMP send.L;                                          03265
               END;                                                  03266
         %font%                                                      03267
            IF NOT format.fds AND NOT format.fdd THEN                03268
               BEGIN                                                 03269
               ↑bp ← ls.rtfont*2 + 1;                                03270
               BUMP send.L;                                          03271
               END;                                                  03272
            END;                                                     03273
         %now send header and string%                                03274
         IF slength > 0 THEN                                         03275
            UNTIL startbp = endbp                                    03276
               OR startbp.bpadr > endbp.bpadr                        03277
               OR (send.L ← send.L +1) >= send.M DO                  03278
               ↑bp ← IF (char ← ↑startbp) = EOL THEN CR ELSE char;
                                                                     03279
         !sout(dspjfn, chbmty + $send, -send.L) ;                    03280
      END;                                                           03281
   ENDCASE                                                           03282
      BEGIN                                                          03283
      r1.sdaid ← da.dahandle;                                        03284
      r1.sstrid ← ls.rtlsid;                                         03285
      save ← r1;                                                     03286
      r2 ← ls.rtbps;                                                 03287
      r3 ← ls.rtbpe;                                                 03288
```

```
        r4.sxcord ← ls.rtx1/256;                                        03289
        r4.sycord ← ((da.dabottom-da.datop) - ls.rty)/256;              03290
        %if default values, use default constants%                      03291
        IF ls.rtfont = da.dafont THEN r4.sfont ← -1                     03292
        ELSE r4.sfont ← ls.rtfont*2 + 1;                                03293
        IF ls.rtcsize = da.dacsize THEN r4.scsize ← -1                  03294
        ELSE r4.scsize ← ls.rtcsize*2 + 1;                              03295
        IF ls.rthinc = da.dacsize THEN r4.shinc ← -1                    03296
        ELSE r4.shinc ← (ls.rthinc/128) + 1;                            03297
        IF NOT SKIP !JSYS strda THEN                                    03298
           IF r1 = syserr THEN NULL                                     03299
           ELSE                                                         03300
              dismes(2,$"NLS Display Error")                            03301
        ELSE ls.rtlsid ← r1 .A 18M;                                     03302
        IF da.dalhandle THEN %linked display area%                      03303
           BEGIN                                                        03304
           r1 ← save;                                                   03305
           r1.sdaid ← da.dalhandle;                                     03306
           IF NOT SKIP !JSYS strda THEN                                 03307
              IF r1 = syserr THEN NULL                                  03308
              ELSE dismes(2, $"Link Display Error");                    03309
           END;                                                         03310
        END;                                                            03311
    RETURN;                                                             03312
    END.
                                                                        03313
%...graphics primitives for alphanumeric (line-processor) displays %
                                                                        02796
    (lpdspstr) PROCEDURE (x, y, string, standoutflag); %display the
    specified string at specified location on a line processor
    alpha-numeric display.  IF x = current then use current position of
    cursor.  If standoutflag is TRUE then cause the string to standout%
                                                                        02797
       LOCAL store, start, char, l, mx;                                 02798
       LOCAL STRING send[200];                                          02799
       REF string;                                                      02800
       IF x NOT= current THEN position(x, y);                           02801
       IF standoutflag THEN standout();                                 02802
       CCPOS SF(*string*);                                              02803
       start ← store ← chbmty + $send;                                  02804
       *send* ← NULL;                                                   02805
       l ← send.L;  mx ← send.M;                                        02806
       LOOP                                                             02807
          BEGIN                                                         02808
          CASE char ← READC OF                                          02809
             = ENDCHR: EXIT LOOP;                                       02810
             = TAB:                                                     02811
                BEGIN                                                   02812
                ↑store ← lptab; BUMP l;  %*send* ← *send*, lptab;%      02813
                END;                                                    02814
             = EOL:                                                     02815
                BEGIN                                                   02816
                !scout(dspjfn, start, -1);                              02817
                *send* ← NULL;                                          02818
                l ← send.L;                                             02819
                store ← start;                                          02820
```

```
            IF x NOT= current THEN position(x ← 0, y ← y+1)
            ELSE
               BEGIN
               !bout(dspjfn, CR);
               !bout(dspjfn, LF);
               pad(lpdlpad);
               END;
            END;
         = LF:
            BEGIN
            !sout(dspjfn, start, -1);
            *send* ← NULL;
            l ← send.L;
            store ← start;
            IF x NOT= current THEN position(x+CCPOS, y)
            ELSE
               BEGIN
               !bout(dspjfn, LF);
               pad(lpdlpad);
               END;
            END;
         ENDCASE
            BEGIN
            ↑store ← char; BUMP l; %*send* ← *send*, char;%
            END;
      IF l >= mx THEN %time to sout%
         BEGIN
         !sout(dspjfn, start, -1);
         *send* ← NULL;
         l ← send.L;
         store ← start;
         END;
      END;
   IF l THEN %output it%
      !sout(dspjfn, start, -1);
   IF standoutflag THEN endstandout();
   track();
   RETURN;
   END.


(pad) PROCEDURE (count);  %pad with count null characters%
   %adjusts for current line speed by dividing count by
   lpbaudfactor%
   count ← MIN((count/lpbaudfactor) + 1, padstr.M);
   IF count > 0 THEN
      !sout(dspjfn, chbmty + $padstr, -count);
   RETURN;
   END.


(position) PROCEDURE % position cursor - alphanumeric displays %

   ( xpos,  % x position %
     ypos); % y position %
   LOCAL STRING send[10];  %for sout%
```

```
   IF xpos NOT IN [0, lpxmax] OR ypos NOT IN [0, lpymax] THEN      02873
      err($"Illegal coordinate detected in POSITION");             02874
   *send* ← lpesc, lpposition, xpos+40B, lpymax-ypos+40B;          02875
   tracking ← FALSE;                                               02876
   !sout(dspjfn, chbmty + $send, -send.L);                         02877
   RETURN;                                                         02878
   END.                                                            02879

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

                                                                   02895
(lprset) PROCEDURE; % reset alphanumeric displays %                02896
   LOCAL save;                                                     02897
   LOCAL STRING send[10];                                          02898
   *send* ← lpesc, lpreset;                                        02899
   save ← tracking := FALSE; %to avoid printer output%             02900
   !sout (dspjfn, chbmty + $send, -send.L);                        02901
   pad(lpdlpad);                                                   02902
   cscreen(); % be sure screen cleared %                           02903
   tracking ← save; %to avoid printer output%                      02904
   RETURN;                                                         02905
   END.                                                            02906

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

                                                                   02918
(track) PROCEDURE; % resume tracking - alphanumeric displays %     02919
   LOCAL STRING send[10];                                          02920
   *send* ← lpesc, lptrack;                                        02921
   !sout(dspjfn, chbmty + $send, -send.L);                         02922
   tracking ← TRUE;                                                02923
```

```
   IF tracksend THEN lppsr( tracksend := 0 );                          02924
   RETURN;                                                             02925
   END.                                                                02926

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

                                                                       02941
(dline) PROCEDURE % delete line - alphanumeric displays %              02942
   (x,      %x coordinate or CURRENT%                                  02943
   y);      %y coordinate or CURRENT%                                  02944
   LOCAL STRING send[10];                                              02945
   IF x NOT= current THEN position(x, y);                              02946
   *send* ← lpesc, lpdline;                                            02947
   !sout(dspjfn, chbmty + $send, -send.L);                             02948
   pad(lpdlpad);                                                       02949
   track();                                                            02950
   RETURN;                                                             02951
   END.                                                                02952

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

                                                                       02967
(lpmarkit) PROCEDURE % bug selection mark - alphanumeric displays %
                                                                       02968
   ( xpos,  % x position of char to bug %                              02969
     ypos); % y position of char to bug %                              02970
   LOCAL save;                                                         02971
   LOCAL STRING send[10];                                              02972
   IF xpos NOT IN [0, lpxmax] OR ypos NOT IN [0, lpymax] THEN          02973
```

```
      err($"Illegal coordinate detected in POSITION");                02974
   *send* ← lpesc, lpbug, xpos+40B, lpymax-ypos+40B;                  02975
   save ← tracking := FALSE; %to avoid printer output%                02976
   !sout(dspjfn, chbmty + $send, -send.L);                            02977
   pad(8);                                                            02978
   tracking ← save; %to avoid printer output%                         02979
   RETURN;                                                            02980
   END.                                                               02981

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

                                                                      02993
(lntter) PROCEDURE; % interrogate alphanumeric displays %             02994
   LOCAL save;                                                        02995
   LOCAL STRING send[10];                                             02996
   *send* ← lpesc, lpintter;                                          02997
   save ← tracking := FALSE; %to avoid printer output%                02998
   !sout(dspjfn, chbmty + $send, -send.L);                            02999
   tracking ← save; %to avoid printer output%                         03000
   RETURN;                                                            03001
   END.                                                               03002

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

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

                                                                      03020
(endstandout) PROCEDURE; % Send end stand out command - alphanumeric
displays %                                                            03021
```

```
   LOCAL STRING send[10];                                              03022
   *send* ← lpesc, lpendstandout;                                      03023
   !sout(dspjfn, chbmty + $send, -send.L);                             03024
   RETURN;                                                             03025
   END.                                                                03026

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

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

                                                                       03041
(lppsr) PROCEDURE % Lineprocessor printer string handler %             03042
   (lppcnt); % number of characters to send %                          03043
   % sends a string of lppcnt chars to Line Processor printer.  Gets
   them from procedure lppch %                                         03044
   LOCAL STRING send[200];                                             03045
   LOCAL store, l;                                                     03046
   IF &lppch=0 THEN err($"no LP character routine - lppsr");           03047
   IF lppcnt<=0 OR lppjfn=0 THEN RETURN;                               03048
   *send* ← NULL;                                                      03049
   l ← send.L;                                                         03050
   store ← chbmty + $send;                                             03051
   ↑store ← lpesc;                                                     03052
   ↑store ← lptptsr;                                                   03053
   ↑store ← 40B;                                                       03054
   ↑store ← lppcnt+40B;                                                03055
   l ← l + 4;                                                          03056
   WHILE (lppcnt ← lppcnt-1) >= 0 DO                                   03057
      BEGIN                                                            03058
      ↑store ← lppch(); % get character and store it %                 03059
      BUMP l;                                                          03060
      END;                                                             03061
   !sout(dspjfn, chbmty+$send, l);                                     03062
   !gtsts(lppjfn);                                                     03063
   IF r2 .A 1B9 THEN % end of file! %                                  03064
      lppterm();                                                       03065
   RETURN;                                                             03066
   END.                                                                03067
(lppterm) PROCEDURE; % terminate LP printing %                         03068
   % aborts Line Processor printer operation %                         03069
   !dti(0); % deassign null %                                          03070
   !dic(4B5,200B); % deactivate chan 28 %                              03071
   lppclose();                                                         03072
   IF NOT SKIP !closf(lppjfn) THEN err(0"cant close text file");
```

```
                                                                        03073
   lppjfn ← tracksend ← 0;                                              03074
   RETURN;                                                              03075
   END.                                                                 03076
                                                                        03077
(lppstart) PROCEDURE % start LP printing from jfn %                     03078
   (jfn, % jfn for text file %                                          03079
   proc); % address of procedure to get chars from %                    03080
   % call this procedure to start printing on LP printer %              03081
   % setup variables %                                                  03082
      lppjfn ← jfn;                                                     03083
      &lppch ← proc;                                                    03084
      tracksend ← lppcol ← lpplin ← 0;                                  03085
   % setup PSI stuff for string requests from LP %                      03086
      chntab[28] ← 1B6+$lppint; % use channel 28 for interrupt %
                                                                        03087
      !ati(0+28); % designate null (zero) as interrupt char %           03088
      !aic(4B5,200B); % activate chn 28 %                               03089
   lppopen(); % send open string %                                      03090
   RETURN;                                                              03091
   END.                                                                 03092
                                                                        03093
(lppint) PROCEDURE; % LP printer interupt routine%                      03094
   %----------------------%                                            03095
                                                                        03096
   %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%                   03097
                                                                        03098
   %----------------------%                                            03099
   %save registers%                                                     03100
   svac1 ← r1;                                                          03101
   r1 ← $svacs;                                                         03102
   !BLT r1, svacse;                                                     03103
   !MOVE r1,40B; sav40 ← r1;                                            03104
   s ← -100; !HRL s,s; !HRRI s,psistk;                                  03105
   m ← s;                                                               03106
   IF tracking THEN lppsr(16)                                           03107
   ELSE tracksend ← tracksend + 16;                                     03108
   %restore registers%                                                  03109
   r1 ← sav40; !MOVEM r1,40B;                                           03110
   !HRLZI r1, svacs;                                                    03111
   !BLT r1, 17B;                                                        03112
   r1 ← svac1;                                                          03113
   !debrk;                                                              03114
   END.
                                                                        03115
```