CEDIT 2

```
< NLS, CEDIT2.NLS.38, >, 22-Apr-78 14:19 JDH ;;;;
FILE cedit2 %  L10  <REL-NLS>cedit2  % % (110,) (rel-nls,cedit2.rel,) %      02
   REGISTER r1=1, r2=2, r3=3, a1=12, a2=13;                                  03
   REF tda;                                                                  04
%print%                                                                      05
   (cprigro)   % UF: core NLS Print Group procedure %
   PROCEDURE (stid1, stid2, vs, da % => statement, integer %);           02668
      % Procedure description                                           02669
         FUNCTION                                                      02670
            none                                                       02671
         ARGUMENTS                                                     02672
            none                                                       02673
         RESULTS                                                       02674
            procedure-value - STID -                                   02675
            2nd-value - INTEGER -                                      02687
         NON-STANDARD CONTROL                                          02676
            none                                                       02677
         GLOBALS                                                       02678
            none                                                       02679
         EXAMPLE                                                       02680
            none                                                       02681
         %                                                             02682
      % Declarations %                                                 02683
         LOCAL stid, cc,tvs1,tvs2,tcacode,tusqcod;                       024
         REF vs, da;                                                     025
      tvs1 _ da.davspec;                                                 026
      tvs2 _ da.davspc2;                                                 027
      tcacode _ da.dacacode;                                             028
      tusqcod _ da.dausqcod;                                             029
      da.davspec _ vs.vs1;                                               030
      da.davspc2 _ vs.vs2;                                               031
      da.dacacode _ vs.vscacode;                                         032
      da.dausqcod _ vs.vsusqcod;                                         033
      stid  _ printg(&da, stid1, stid2, groupv, 0 : cc);                 034
      da.davspec _ tvs1;                                                 035
      da.davspc2 _ tvs2;                                                 036
      da.dacacode _ tcacode;                                             037
      da.dausqcod _ tusqcod;                                             038
      RETURN (stid, cc);                                                 039
      END.   %%
```

```
                                                                        02686
(Cprijou)   % UF: core NLS Print Journal procedure %
PROCEDURE (stid, da % => no value %);                                   02688
   % Procedure description                                             02689
      FUNCTION                                                          02690
         none                                                           02691
      ARGUMENTS                                                         02692
         none                                                           02693
      RESULTS                                                           02694
         none                                                           02695
      NON-STANDARD CONTROL                                              02696
         none                                                           02697
      GLOBALS                                                           02698
         none                                                           02699
      EXAMPLE                                                           02700
         none                                                           02701
      %                                                                 02702
   % Declarations %                                                    02703
      LOCAL pjb, pjba, pjbb, pjbc;                                        043
         %may delete this statement and make pjb a formal parameter%      044
      LOCAL pjl, pjbrad, pjvspec;                                         045
      LOCAL pjbflag %journal branch found%, pjoflag %mail found%;         046
      LOCAL jstid, sw;                                                    047
      REF pjb, da, sw;                                                    048
   pjbflag _ pjoflag _ 0;                                                 049
   %may delete these when pjb is made a calling parameter%                050
      &pjb _ $pjba;                                                       051
      pjba_$"Journal"; % pjbb_$"Info"; pjbc_$"Action"; %                  052
      pjl_1; %used to be 3 when "Info" and "Action" weren't commented out
      above%                                                              053
   pjvspec _ da.davspec;                                                  054
   pjvspec.vsusgf _ TRUE;                                                 055
   WHILE (pjl_pjl-1) >= 0 DO IF (pjbrad _ pjb[pjl]) THEN                  056
      BEGIN                                                               057
      IF (jstid _ namelook (stid, pjbrad) )= endfil THEN REPEAT LOOP;     058
      pjbflag _ TRUE;                                                     059
      IF getsub(jstid)= jstid THEN REPEAT LOOP;                           060
      pjoflag _ TRUE;                                                     061
      &sw _ openseq( jstid, jstid, pjvspec, da.davspc2, $pjseqg, 0);      062
      printg(&da, jstid, jstid, groupv, &sw);                             063
      closeseq(&sw);                                                      064
      END;                                                                065
   IF NOT pjbflag THEN err($" No Journal Branches");                      066
   IF NOT pjoflag THEN err($"No mail");                                   067
   RETURN;                                                                068
   END.   %%
```

```
                                                                    02706
(cprires)   % UF: core NLS Print Rest procedure %
PROCEDURE (stid, da % => statement, integer %);                     02707
   % Procedure description                                          02708
      FUNCTION                                                      02709
         none                                                       02710
      ARGUMENTS                                                     02711
         none                                                       02712
      RESULTS                                                       02713
         procedure-value - STID -                                   02726
         2nd-value - INTEGER -                                      02727
      NON-STANDARD CONTROL                                          02715
         none                                                       02716
      GLOBALS                                                       02717
         none                                                       02718
      EXAMPLE                                                       02719
         none                                                       02720
      %                                                             02721
   % Declarations %                                                02722
      LOCAL cc;                                                       08
      REF da;                                                         09
   stid _ printg( &da, stid, endfil, groupv, 0 : cc );               010
   RETURN(stid, cc);                                                 011
   END.   %%
```

```
                                                                    02725
(cprista)   % UF: core NLS Print Statement procedure %
PROCEDURE (stid, vs, da % => statement, integer %);                 02728
   % Procedure description                                          02729
      FUNCTION                                                      02730
         none                                                       02731
      ARGUMENTS                                                     02732
         none                                                       02733
      RESULTS                                                       02734
         procedure-value - STID -                                   02748
         2nd-value - INTEGER -                                      02749
      NON-STANDARD CONTROL                                          02736
         none                                                       02737
      GLOBALS                                                       02738
         none                                                       02739
      EXAMPLE                                                       02740
         none                                                       02741
      %                                                             02742
   % Declarations %                                                02743
      LOCAL cc;                                                       015
      REF vs, da;                                                     016
   da.davspec _ vs;                                                   017
   da.davspc2 _ vs[1];                                                018
   stid _ printg(&da, stid, stid, stmtv, 0 : cc);                     019
   RETURN (stid, cc);                                                 020
   END.   %%
```

```
                                                                        02746
%protect%                                                                070
   (cprofil)   % GB: set protection of a group of files and PCs %
   PROCEDURE (rhstn, fname, mask, prot, astr % => no value %);          02133
      % Procedure description                                          02134
         FUNCTION                                                      02135
            Sets the protection of a group of files and their partial copies.
            (Check the TENEX JSYS Manual for details on file protection.)  The
            names of the files changed are put in 'astr'.              02136
         ARGUMENTS                                                     02137
            rhstn - INTEGER -                                          03211
               number of the host on which the files reside.  Only the local
               host ('lhostn') is currently implemented.               03212
            fname - STRING -                                           03213
               file group name string in TENEX format.  Connected directory is
               assumed if a directory is not specified.  Examples:     03214
                  "*.*"                                                03215
                  "<SMITH>*.PC"                                        03216
                  "<SMITH>FOO.NLS;3"                                   03217
            mask - INTEGER - mask indicating which bits to change      02154
            prot - INTEGER - new protection bits                       02155
               77    77     77                                         03219
               self  group  world                                      03220
                  77 = (read, write, execute, append, *, unused)       03221
                     * = access determined by individual pages in the file
                                                                        03222
               Example: 400000 = nobody but me can do anything to the file,
               and I can only read it (pretty safe!)                   03223
               Example: 777752 = the normal NLS setting                03224
            astr - STRING - string in which to put messages            03218
         RESULTS                                                       02139
            none                                                       02140
         NON-STANDARD CONTROL                                          02141
            Error if                                                   02142
               host is not the local host                              03225
               a file doesn't exist or can't be changed                03226
               not a disk file                                         03227
         GLOBALS                                                       02143
            <lots>                                                     02144
         EXAMPLE                                                       02145
            cprofil(lhostn, $"<SMITH>*.NLS", 777777B, 775200B, $string)  02146
               Meaning: I can do anything, my group can read and execute,
               others can do nothing (not even get a directory listing of it).
                                                                        03228
            %                                                          02147
      % Declarations %                                                 02148
         LOCAL                                                         079
            fjfn,        % main jfn %                                   080
            jfnflgs,     % left half flags for a group jfn %            081
            pcjfn,       % jfn of partial copy %                        082
            flags;       % bits indicating * typed for file name fields %  083
         LOCAL STRING                                                  085
            uname[200],          % complete name as entered by user %   086
            udirname[40],        % user input directory name %          087
            ufilname[40],        % user input file name %               088
            uextname[40],        % user input extension name %          089
```

```
        uverno[40],          % user input version number %                    090
        ulftovr[40],         % user input beyond version number %             091
        jfnname[200],        % complete name from jfns %                      092
        errstring[200],      % eror message string %                          093
        tstring[200];        % temporary string %                             094
    REF fname, astr;                                                          096
% initial variables %                                                        098
    fjfn _ jfnflgs _ pcjfn _ flags _ 0;                                       099
% find out if remote or local (disk) file %                                 0100
    CASE rhstn OF                                                            0101
       = lhostn:  NULL;                                                      0102
       ENDCASE err( $"remote file manipulations not implemented yet" );
                                                                             0103
% now parse user input strings %                                            0104
    parseinput( &fname, $udirname, $ufilname, $uextname, $uverno,
    $ulftovr, $uname, $flags);                                               0105
% get main jfn (old file only, group jfn) %                                 0106
    IF NOT (fjfn _ sgtjfn( gtjoif .V gtjstr + 0, $uname, $errstring :
    jfnflgs ) )THEN err( $errstring );                                       0107
% now find out if we support this type of file %                            0108
    IF NOT chkdev( fjfn ) THEN err( $"only disk files supported" );          0109
% now loop to get all files in the group %                                  0110
    LOOP                                                                     0111
       BEGIN                                                                 0112
       % check to make sure this not a PC with * for extension name %
                                                                             0113
          IF NOT chkpcs( fjfn, flags) THEN GOTO gpronxjfn;                   0114
       % get actual file name into string for use later (maybe) %           0115
          r3 _ 011110B6    %  directory file ext ver %                       0116
             + (IF jfnflgs.lhjb9  THEN 1B6 ELSE 0) % ;Protection %           0117
             + (IF jfnflgs.lhjb10 THEN 1B5 ELSE 0) % ;Account %              0118
             + (IF jfnflgs.lhjb11 THEN 4B4 ELSE 0) % ;T %                    0119
             + 1;              % with proper file punctuation %             0120
          jfnflink( fjfn, $jfnname, r3);                                     0121
       % get PC jfn and find out if this user can access this PC %          0122
          IF NOT getpcjfn( fjfn, $pcjfn, $errstring, FALSE) THEN             0123
             BEGIN % this user can't access this file %                      0124
             *tstring* _ *jfnname*, " protection not changed because ",
             *errstring*;                                                    0125
             IF flags THEN                                                   0126
                BEGIN                                                        0127
                *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;            0128
                GOTO gpronxjfn;                                              0129
                END                                                          0130
             ELSE                                                            0131
                BEGIN                                                        0132
                IF NOT SKIP !rljfn( fjfn) THEN NULL;                         0133
                err( $tstring );                                             0134
                END);                                                        0135
             END;                                                            0136
       % now protect the file (and its partial copy) %                       0137
          IF NOT proflandpc( fjfn, pcjfn, mask, prot, $errstring) THEN
                                                                             0138
             BEGIN % can't delete file (or partial copy) %                   0139
             *tstring* _ *jfnname*, " protection not changed because ",
             *errstring*;                                                    0140
```

```
            IF flags THEN                                                   0141
               BEGIN                                                        0142
               *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;            0143
               GOTO gpronxjfn;                                              0144
               END                                                          0145
            ELSE                                                            0146
               BEGIN                                                        0147
               IF NOT SKIP !rljfn( fjfn) THEN NULL;                         0148
               IF NOT SKIP !rljfn( pcjfn) THEN NULL;                        0149
               err( $tstring );                                             0150
               END;                                                         0151
            END);                                                           0152
         % now tell the user we have protected this file %                  0153
            *astr* _ *astr*, "   ", *jfnname*;                              0154
            IF pcjfn THEN *astr* _ *astr*, " and its partial copy";         0155
            *astr* _ *astr*, CR, LF;                                        0156
         % now get rid of the jfn for the partial %                         0157
            IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                          0158
         % now get the next file in the group %                             0159
            (gpronxjfn):                                                    0160
            r1.LH _ jfnflgs;                                                0161
            r1.RH _ fjfn;                                                   0162
            IF NOT SKIP !gnjfn( r1 ) THEN EXIT LOOP;                        0163
         END;                                                               0164
   % now get rid of any lingering jfns %                                    0165
      IF NOT SKIP !rljfn( fjfn ) THEN NULL;                                 0166
      IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                                0167
   % all done now, so return %                                             0168
      RETURN;                                                               0169
   END.   %%
```

```
                                                                        02151
%renumber%                                                               0174
   (crensidfil)   % GB: core NLS Renumber SID's in File procedure %
   PROCEDURE (fileno % => no value %);                                  02361
      % Procedure description                                          02362
         FUNCTION                                                       02363
            Puts all SIDs in the file in sequential order starting with 01 for
            the origin statement.                                       02364
         ARGUMENTS                                                      02365
            fileno - INTEGER - NLS file number of the file              02366
         RESULTS                                                        02367
            none                                                        02368
         NON-STANDARD CONTROL                                           02369
            Error if the file number is not a legal file                02370
         GLOBALS                                                        02371
            sidcnt, filhed, filehead, orgstid - read only               02372
         EXAMPLE                                                        02373
            crensidfil(stid.stfile)                                     02374
         %                                                              02375
      % Declarations %                                                  02376
         LOCAL stid, sidc;                                              0177
         REF sidc;                                                      02380
      IF fileno NOT IN [1,filcnt] THEN err($"Illegal file");            0178
      &sidc _ ($sidcnt-$filhed) + filehead[fileno];                     0179
      sidc _ 0;                                                         0180
      stid _ orgstid;  stid.stfile _ fileno;                            0181
      rensorg(stid, &sidc);                                            03174
      RETURN;                                                           0184
      END.   %%
```

```
                                                                        02379
(rensorg)                                                               03170
   %renumbers sid's in branch hanging from origin stid%                 03172
      %sidc is address of sid counter%                                  03173
   PROCEDURE (stid, sidc);                                              03171
   LOCAL curpty, sdb, infstid;                                          03175
   REF sidc, sdb;                                                       03176
   DO                                                                   03177
      BEGIN                                                             03178
      stosid(stid, sidc _ sidc + 1);                                    03179
      IF NOT getorf(stid) THEN                                          03180
         BEGIN                                                          03181
         curpty _ getsdb(stid); %first property%                        03182
         LOOP                                                           03183
            BEGIN                                                       03184
            IF curpty.stpsdb=0 THEN EXIT;                               03185
            lodent(curpty, sdbtyp : &sdb);                              03186
            curpty.stpsdb _ sdb.spsdb;                                  03187
            CASE sdb.sptype OF                                          03188
               =dhtyp, =chtyp: IF (infstid _ sdb.sitpsid) THEN          03189
                  BEGIN                                                 03190
                  infstid.stfile _ stid.stfile;                         03191
                  rensorg (infstid, &sidc);                             03192
                  END;                                                  03193
               ENDCASE NULL;                                            03194
            END;                                                        03195
         END;                                                           03196
      END UNTIL (stid _ getnxt(stid)) = endfil;                         03197
   RETURN; END.   %%
```

```
                                                                        03198
%replace%                                                               0186
   (crepgro)   % GB: core NLS Replace Group procedure %
   PROCEDURE (bugdit, atbug1, atbug2, bybug1, bybug2 % => statement, statement
   %);                                                                  02248
      % Procedure description                                          02249
         FUNCTION                                                       02250
            Replaces the group at 'atbug1' through 'atbug2' with the group at
            'bybug1' through 'bybug2'.  'Bugdit' tells whether the replacement
            is a group of existing statements or a string to make into a
            statement.                                                  02251
         ARGUMENTS                                                      02252
            bugdit - BOOLEAN - "bugged it"                              02267
               TRUE  => the replacement is an existing group            02268
               FALSE => the replacement is a string to make into a statement
                                                                        02270
            atbug1 - TEXT POINTER - start of group to replace           02273
            atbug2 - TEXT POINTER - end of group to replace             02278
            bybug1 - TEXT POINTER -                                     02274
               start of replacement group or string                     02275
            bybug2 - TEXT POINTER -                                     02276
               end of replacement group or string                       02277
         RESULTS                                                        02254
            procedure-value - STID - head of the new group              02255
            2nd-result - STID - tail of the new group                   03229
         NON-STANDARD CONTROL                                           02256
            none                                                        02257
         GLOBALS                                                        02258
            levsuc - read only                                          02259
         EXAMPLE                                                        02260
            head _ crepgro(TRUE, $tp1, $tp2, $tp3, $tp4 : tail)        02261
         %                                                              02262
      % Declarations %                                                  02263
         LOCAL head, tail;                                               0201
         REF atbug1, atbug2, bybug1, bybug2;                             0202
        %msntx();%                                                      03435
      atbug1 _ grptst(atbug1, atbug2 :atbug2);                          03436
      IF bugdit THEN                                                    03437
         BEGIN                                                          03438
         bybug1 _ grptst(bybug1, bybug2 :bybug2);                       03439
         head _ copgrp(atbug2, levsuc, bybug1, bybug2, FALSE: tail);    03440
         cdelgro(atbug1, atbug2, FALSE, 0);                             03441
         END                                                            03442
      ELSE head _ tail _ rpllit(atbug1, atbug2, &bybug1, &bybug2);      03443
        %msftx();%                                                      03444
      RETURN(head, tail);                                               03445
      END.
                                                                        03446
   (crepsta)   % GB: core NLS Replace Statement procedure %
   PROCEDURE (bugdit, atbug, bybug1, bybug2 % => statement %);          02216
      % Procedure description                                          02217
         FUNCTION                                                       02218
            Replaces the statement at 'atbug' with another statement or a
            string.  The SID and statement number of the replaced statement
            are not changed.  The name delimiters are also not changed,
            unless the replacement is another statement, in which case the new
```

```
      statement's name delimiters are used.                                  02219
   ARGUMENTS                                                                 02220
      bugdit - BOOLEAN - "bugged it"                                         02221
         TRUE => the replacement is an existing statement                    02238
            (at bybug1)                                                      02241
         FALSE => the replacement is a string                                02239
            (at bybug1 bybug2)                                               02242
         to make into a statement                                            02243
      atbug - TEXT POINTER - statement to replace                            02237
      bybug1 - TEXT POINTER -                                                02235
         replacement statement or start of string                            02247
      bybug2 - TEXT POINTER -                                                02245
         end of string; not used if replacement is a statement               02246
   RESULTS                                                                   02222
      procedure-value - STID -                                               02223
         the new statement; this will have the same stid as the replaced
         statement                                                           02244
   NON-STANDARD CONTROL                                                      02224
      none                                                                   02225
   GLOBALS                                                                   02226
      none                                                                   02227
   EXAMPLE                                                                   02228
      stid _ crepsta(TRUE, $tp1, $tp2, 0)                                   02229
   %                                                                         02230
% Declarations %                                                            02231
   LOCAL stid;                                                              0190
   REF atbug, bybug1, bybug2;                                               0191
treplist(atbug);                                                            0192
IF bugdit THEN                                                              0193
   copplist(bybug1, atbug)                                                  0194
ELSE %literal input%                                                        0195
   ST atbug _ bybug1 bybug2;                                                0196
RETURN(atbug);                                                              0197
END.  %%
```

```
                                                                        02234
(creptex)   % GB: core NLS Replace Text procedure %
PROCEDURE (bug1, bug2, bug3, bug4 % => no value %);                     02279
   % Procedure description                                              02280
      FUNCTION                                                          02281
         Replaces the text at 'bug1' through 'bug2' with the text at 'bug3'
         through 'bug4'.  Either text may be in a statement or a string.
                                                                        02282
      ARGUMENTS                                                         02283
         bug1 - TEXT POINTER - start of text to replace                 02284
         bug2 - TEXT POINTER - end of text to replace                   02298
         bug3 - TEXT POINTER - start of replacement text                02299
         bug4 - TEXT POINTER - end of replacement text                  02300
      RESULTS                                                           02285
         none                                                           02286
      NON-STANDARD CONTROL                                              02287
         none                                                           02288
      GLOBALS                                                           02289
         none                                                           02290
      EXAMPLE                                                           02291
         creptex($tp1, $tp2, $tp3, $tp4)                                02292
      %                                                                 02293
   % Declarations %                                                     02294
      REF bug1, bug2, bug3, bug4;                                        0231
     %msntx();%                                                          0232
   cldtxt(&bug1, &bug2);                                                0233
   ST bug1 _ SF(bug1) bug1,                                             0234
      $bug3 bug4,     % $ => don't move markers %                       03681
      bug2 SE(bug1);                                                    03682
     %msftx();%                                                         0235
   RETURN;                                                              0236
   END.   %%
```

```
                                                                        02297
(rplilt)   % LB: replaces a group with a literal string %
PROCEDURE (grp1, grp2, t1, t2 % => statement %);                        02750
   % Procedure description                                              02751
      FUNCTION                                                          02752
         Replaces the group defined by the first two arguments passed it
         with a new statement containing the text in the string passed it.
         It returns the value of the new stid.                          02769
         First it gets a new stid, and inserts it as the successor of GRP2.
         It then deletes the group bounded by GRP1, GRP2 and puts the text
         in the string passed it in the new SDB.                        02770
      ARGUMENTS                                                         02754
         none                                                           02755
      RESULTS                                                           02756
         procedure-value - STID - new statement created                 02757
      NON-STANDARD CONTROL                                              02758
         none                                                           02759
      GLOBALS                                                           02760
         none                                                           02761
      EXAMPLE                                                           02762
         none                                                           02763
      %                                                                 02764
   % Declarations %                                                     02765
      LOCAL stid; %stid for new statement%                               0220
      REF t1, t2;                                                        0221
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");              0222
   stid _ newrng(grp1.stfile);                                           0223
   inss(grp2, stid, stid);                                               0224
   cdelgro(grp1, grp2, FALSE, 0);                                        0225
   ST stid _ t1 t2;                                                      0226
   RETURN(stid);                                                         0227
   END.   %%
```

```
                                                                        02768
   %routines for replace and transpose number%                           0238
      (trnchk) PROCEDURE (ptr1,ptr2,ptr3,ptr4);                          0239
         %moves pointers around for transpose number%                    0240
         LOCAL delt;                                                     0241
         delt _ [ptr2+1] - [ptr1+1] - [ptr4+1] + [ptr3+1];               0242
         CASE delt OF                                                    0243
            < 0: rjshift(ptr1,delt);                                     0244
            > 0: rjshift(ptr3,-delt);                                    0245
            ENDCASE;                                                     0246
         RETURN;                                                         0247
         END.                                                            0248
                                                                         0249
      (rjshift) PROCEDURE (ptr,delt);                                    0250
         REF ptr;                                                        0251
         FIND ptr <;                                                     0252
         LOOP                                                            0253
            IF (delt := delt+1) >= 0 OR                                  0254
            (NOT FIND SP ^ptr) THEN RETURN;                              0255
         END.                                                            0256
                                                                         0257
      (rjrepn) PROCEDURE (ptr1, ptr2, astring);                          0258
         LOCAL delt;                                                     0259
         REF astring;                                                    0260
         delt _ [ptr2+1] - [ptr1+1] - astring.L;                         0261
         CASE delt OF                                                    0262
            < 0: rjshift(ptr1,delt);                                     0263
            > 0: BEGIN                                                   0264
               *num* _ *astring*;                                        0265
               *astring* _ NULL;                                         0266
               UNTIL (delt _ delt-1) < 0 DO *astring* _ *astring*,       0267
               SP;                                                       0268
               *astring* _ *astring*, *num*;                             0269
               END;                                                      0270
            ENDCASE;                                                     0271
         RETURN;                                                         0272
         END.                                                            0273
                                                                         0274
%reset%                                                                  0275
   (cresarcfil)   % UB: core NLS Reset Archive File request procedure %
   PROCEDURE (rhostn, fname % => no value %);                            02771
      % Procedure description                                           02772
         FUNCTION                                                        02773
            Not implemented yet.                                         02774
         ARGUMENTS                                                       02775
            none                                                         02776
         RESULTS                                                         02777
            none                                                         02778
         NON-STANDARD CONTROL                                            02779
            none                                                         02780
         GLOBALS                                                         02781
            none                                                         02782
         EXAMPLE                                                         02783
            none                                                         02784
         %                                                               02785
      % Declarations %                                                   02786
```

```
    REF fname;                                                                  0278
  CASE rhostn OF                                                                02794
    = lhostn: err( notyet );                                                    02795
    ENDCASE err($"Remote File Manipulations Not Implemented Yet");              02796
  RETURN;                                                                       02797
  END.   %%
```

```
                                                                        02789
(crescasmod)   % UB: core NLS Reset Case Mode procedure %
PROCEDURE %( => no value )%;                                            02478
   % Procedure description                                             02479
      FUNCTION                                                          02480
         Sets the default mode for the BASE Force Case command to upper
         case.                                                          02481
      ARGUMENTS                                                         02482
         none                                                           02483
      RESULTS                                                           02484
         none                                                           02485
      NON-STANDARD CONTROL                                              02486
         none                                                           02487
      GLOBALS                                                           02488
         xsmode - CHANGED - default case mode for Force Case command    02489
         upcase - read only - upper case                                03230
      EXAMPLE                                                           02490
         none                                                           02491
      %                                                                 02492
   % Declarations %                                                    02493
   xsmode _ upcase;                                                      0286
   RETURN;                                                               0287
   END.   %%
```

```
                                                                        02496
(creslindef)   % UB: core NLS Reset Link Default for file procedure %
PROCEDURE (fileno, f1, f2 % => no value %);                             02157
   % Procedure description                                              02158
      FUNCTION                                                          02159
         Not implemented yet.                                           02160
         Login or connected directory??                                  0291
      ARGUMENTS                                                         02161
         none                                                           02162
      RESULTS                                                           02163
         none                                                           02164
      NON-STANDARD CONTROL                                              02165
         none                                                           02166
      GLOBALS                                                           02167
         none                                                           02168
      EXAMPLE                                                           02169
         none                                                           02170
      %                                                                 02171
   % Declarations %                                                     02172
   err( notyet );                                                        0292
   RETURN;                                                               0293
   END.   %%
```

```
                                                                        02175
(cresngro)   % GB: core NLS Reset Name delimiters Group procedure %
PROCEDURE (stid1, stid2, da % => no value %);                           02535
   % Procedure description                                              02536
      FUNCTION                                                          02537
         Sets the name deliniters in all statements in the group at 'stid1'
         through 'stid2' to the default delimiters specified by the User
         Options.  Only those statements are changed which pass the
         viewspecs (including content filter, if any) in the display area
         specified.                                                     02538
      ARGUMENTS                                                         02539
         stid1 - STID - head of group to change                         02540
         stid2 - STID - tail of group to change                         03231
         da - DISPLAY AREA - for viewspecs                              03232
      RESULTS                                                           02541
         none                                                           02542
      NON-STANDARD CONTROL                                              02543
         none                                                           02544
      GLOBALS                                                           02545
         dfnmd1, dfnmdr - read only - default name delimiters           02546
      EXAMPLE                                                           02547
         cresngro(stid1, stid2, lda())                                  02548
      %                                                                 02549
   % Declarations %                                                     02550
      REF da;                                                            0297
   csetngro(stid1, stid2, dfnmdl, dfnmdr, &da);                          0298
   RETURN;                                                               0299
   END.   %%
```

```
                                                                        02553
(cresnsta)   % GB: core NLS Reset Name delimiters Statement proc %
PROCEDURE (stid % => no value %);                                       02554
   % Procedure description                                             02555
      FUNCTION                                                          02556
         Sets the name delimiters in the statement at 'stid' to the default
         delimiters specified by the User Options.                      02557
      ARGUMENTS                                                         02558
         stid - SFID - statement to change                              03448
      RESULTS                                                           02560
         none                                                           02561
      NON-STANDARD CONTROL                                              02562
         none                                                           02563
      GLOBALS                                                           02564
         dfnmdl, dfnmdr - read only - default name delimiters           03451
      EXAMPLE                                                           02566
         cresnsta(stid, lda())                                          02567
      %                                                                 02568
   % Declarations %                                                     02569
      REF da;                                                           0303
   csetnsta(stid, stid, dfnmdl, dfnmdr);                                0304
   RETURN;                                                              0305
   END.   %%
```

```
                                                                        02572
(crestemmod)   % UB: core NLS Reset Temporary Modifications proc %
PROCEDURE (fileno, lock % => no value %);                               02799
   % Procedure description                                              02800
      FUNCTION                                                          02801
         none                                                           02802
      ARGUMENTS                                                         02803
         none                                                           02804
      RESULTS                                                           02805
         none                                                           02806
      NON-STANDARD CONTROL                                              02807
         none                                                           02808
      GLOBALS                                                           02809
         none                                                           02810
      EXAMPLE                                                           02811
         none                                                           02812
      %                                                                 02813
   % Declarations %                                                     02814
      LOCAL fl;                                                          0309
      LOCAL STRING filestr[100];                                         0310
      REF fl;                                                            0311
   &fl _ flntadr(fileno);                                                0312
   IF lock THEN                                                          0313
      IF NOT lockfile(fileno) THEN RETURN                                0314
      ELSE NULL                                                          0315
   ELSE                                                                  0316
      BEGIN                                                              0317
      IF NOT fl.flbrws THEN                                              0318
         BEGIN                                                           0319
         dismes(2, $"This File not in Temporary Modifications Mode");    0320
         RETURN;                                                         0321
         END;                                                            0322
      unlkfile(fileno, FALSE);                                           0323
      END;                                                               0324
   fl.flbrws _ FALSE;                                                    0325
   RETURN;                                                               0326
   END.   %%
```

```
                                                                        02817
%retrieve%                                                               0328
   (cretarcfil)   % UB: core NLS Retrieve Archived File procedure %
   PROCEDURE (rhostn, fname % => no value %);                           02818
      % Procedure description                                          02819
         FUNCTION                                                       02820
            Not implemented yet.                                        02821
         ARGUMENTS                                                      02822
            none                                                        02823
         RESULTS                                                        02824
            none                                                        02825
         NON-STANDARD CONTROL                                           02826
            none                                                        02827
         GLOBALS                                                        02828
            none                                                        02829
         EXAMPLE                                                        02830
            none                                                        02831
         %                                                              02832
      % Declarations %                                                 02833
         REF fname;                                                     0331
      CASE rhostn OF                                                    0332
         = lhostn: err( notyet );                                       0333
         ENDCASE err($"Remote File Manipulations Not Implemented Yet"); 0334
      RETURN;                                                           0335
      END.   %%
```

```
                                                                        02836
%set%                                                                    0337
   (csetcgro)   % GB: core NLS Set Case Group procedure %
   PROCEDURE (stid1, stid2, type % => no value %);                      02421
      % Procedure description                                          02422
        FUNCTION                                                       02423
           Changes the case of all characters in the group at 'stid1' through
           'stid2' to the case specified by 'type'.                    02424
        ARGUMENTS                                                      02425
           stid1 - STID - head of group to change                      03452
           stid2 - STID - tail of group to change                      03453
           type -INTEGER - case to make the characters                 03454
              upcase  => upper case                                    03455
              lowcase => lower case                                    03456
              iupcase => first letter in every word upper case, all others
              lower case                                               03457
        RESULTS                                                        02427
           none                                                        02428
        NON-STANDARD CONTROL                                           02429
           none                                                        02430
        GLOBALS                                                        02431
           none                                                        02432
        EXAMPLE                                                        02433
           csetcgro(stid1, stid2, upcase)                              02434
        %                                                              02435
      % Declarations %                                                 02436
        %msntx();%                                                      0340
      stid1 _ grptst(stid1, stid2 :stid2);                              0341
      stid2 _ getend(stid2);                                            0342
      IF stid1 # stid2 THEN                                             0343
         DO csetcsta(stid1, type) UNTIL (stid1 _ getnxt(stid1)) = stid2;  0344
      csetcsta(stid2, type);                                            0345
        %msftx();%                                                      0346
      RETURN;                                                           0347
      END.   %%
```

```
                                                                        02439
(csetcmod)   % UB: core NLS Set Case Mode procedure %
PROCEDURE (type % => no value %);                                       02440
   % Procedure description                                             02441
      FUNCTION                                                          02442
         Sets the default mode for the BASE Force Case command.         02443
      ARGUMENTS                                                         02444
         type -INTEGER - case to make the characters                    03460
            upcase  => upper case                                       03461
            lowcase => lower case                                       03462
            iupcase => first letter in every word upper case, all others
            lower case                                                  03463
      RESULTS                                                           02446
         none                                                           02447
      NON-STANDARD CONTROL                                              02448
         Error if type is not one of upcase, lowcase, iupcase           02449
      GLOBALS                                                           02450
         xsmode - CHANGED - default case mode for Force Case command    03464
         upcase, lowcase, iupcase - read only - case modes              02451
      EXAMPLE                                                           02452
         csetcmod(iupcase)                                              02453
      %                                                                 02454
   % Declarations %                                                     02455
   xsmode _ (CASE type OF                                               0351
      = upcase, = lowcase, = iupcase: type;                             0352
      ENDCASE err($"Illegal case mode"));                               0353
   RETURN;                                                              0354
   END.   %%
```

```
                                                                        02458
(csetcsta)   % GB: core NLS Set Case Statement procedure %
PROCEDURE (stid, type % => no value %);                                 02459
   % Procedure description                                             02460
      FUNCTION                                                          02461
         Changes the case of all characters in the statement at 'stid' to
         the case specified by 'type'.                                  02462
      ARGUMENTS                                                         02463
         stid - STID - statement to change                              03465
         type -INTEGER - case to make the characters                    03467
            upcase  => upper case                                       03468
            lowcase => lower case                                       03469
            iupcase => first letter in every word upper case, all others
            lower case                                                  03470
      RESULTS                                                           02465
         none                                                           02466
      NON-STANDARD CONTROL                                              02467
         Error if type is not one of upcase, lowcase, iupcase           02468
      GLOBALS                                                           02469
         upcase, lowcase, iupcase - read only - case modes              03472
      EXAMPLE                                                           02471
         csetcsta(stid, upcase)                                         02472
      %                                                                 02473
   % Declarations %                                                    02474
      LOCAL TEXT POINTER ptr1, ptr2;                                     0358
     %msntx();%                                                          0359
   FIND SF(stid) ^ptr1 SE(stid) ^ptr2;                                   0360
   CASE type OF                                                          0361
      =upcase: % capital %                                               0362
         ST stid _ +ptr1 ptr2;                                           0363
      =lowcase: % lower %                                                0364
         ST stid _ -ptr1 ptr2;                                           0365
      =iupcase: % initial upper %                                        0366
         BEGIN                                                           0367
         iupper( $ptr1, $ptr2 );                                         0368
         END;                                                            0369
      ENDCASE  err($"Illegal Case Mode");                                0370
   %msttx();%                                                            0371
   RETURN;                                                               0372
   END.   %%
```

```
                                                                        02477
(csetctex)   % GB: core NLS Set Case Text procedure %
PROCEDURE (bug1, bug2, type % => no value %);                           02400
   % Procedure description                                              02401
      FUNCTION                                                          03528
         Changes the case of all characters in the text at 'bug1' through
         'bug2' to the case specified by 'type'.                        03529
      ARGUMENTS                                                         03530
         bug1 - STID - start of text to change                          03531
         bug2 - STID - end of text to change                            03532
         type -INTEGER - case to make the characters                    03533
            upcase  => upper case                                       03534
            lowcase => lower case                                       03535
            iupcase => first letter in every word upper case, all others
            lower case                                                  03536
      RESULTS                                                           02406
         none                                                           02407
      NON-STANDARD CONTROL                                              02408
         Error if type is not one of upcase, lowcase, iupcase           02409
      GLOBALS                                                           02410
         upcase, lowcase, iupcase - read only - case modes              03537
      EXAMPLE                                                           02412
         csetctex($tp1, $tp2, lowcase)                                  02413
      %                                                                 02414
   % Declarations %                                                     02415
      REF bug1, bug2;                                                    0400
     %msntx();%                                                          0401
   CASE type OF                                                         0402
      =upcase: % capital %                                              0403
         ST bug1 _ SF(bug1) bug1, +bug1 bug2, bug2 SE(bug1);            0404
      =lowcase: % lower %                                               0405
         ST bug1 _ SF(bug1) bug1, -bug1 bug2, bug2 SE(bug1);            0406
      =iupcase: % initial upper %                                       0407
         iupper(&bug1, &bug2);                                          0408
      ENDCASE err($"Illegal Case Mode");                                0409
     %msttx();%                                                         0410
   RETURN;                                                              0411
   END.   %%
```

```
                                                                        02418
(csetextname)   % UB: core NLS Set External Names link file name %
PROCEDURE (fileno, adstr % => no value %);                              02837
   % Procedure description                                              02838
      FUNCTION                                                          02839
         none                                                           02840
      ARGUMENTS                                                         02841
         none                                                           02842
      RESULTS                                                           02843
         none                                                           02844
      NON-STANDARD CONTROL                                              02845
         none                                                           02846
      GLOBALS                                                           02847
         none                                                           02848
      EXAMPLE                                                           02849
         none                                                           02850
      %                                                                 02851
   % Declarations %                                                     02852
      LOCAL TEXT POINTER stid, tps, tp1, tp2, tpe1, tpe2;               0460
      LOCAL ldstr[40];                                                  0461
      REF adstr;                                                        0462
   stid _ origin;  stid.stfile _ fileno;  stid[1] _ 1;                  0464
   tpe1 _ adstr[ls];  tpe1[1] _ adstr[ls+1];                            0465
   tpe2 _ adstr[le];  tpe2[1] _ adstr[le+1];                            0466
   IF FIND SF(stid) ["EXTERNAL LINKS:"] $(SP/TAB/CR/LF/EOL) ^tps THEN   0467
      BEGIN                                                             0468
      ON SIGNAL ELSE                                                    0469
         BEGIN                                                          0470
         ON SIGNAL ELSE;                                                0471
         GOTO noenlf;                                                   0472
         END;                                                           0473
      inkprs( $tps, $ldstr );                                           0474
      ON SIGNAL ELSE;                                                   0475
      tp1 _ ldstr[ls];  tp1[1] _ ldstr[ls+1];                           0476
      tp2 _ ldstr[le];  tp2[1] _ ldstr[le+1];                           0477
      IF tp1[1] = tps[1] THEN                                           0478
         ST stid _ SF(stid) tp1, tpe1 tpe2, tp2 SE(stid)                0479
      ELSE                                                              0480
         (noenlf): ST stid _ SF(stid) tps, tpe1 tpe2, tps SE(stid);     0481
      END                                                               0482
   ELSE ST stid _ SF(stid) SE(stid), "; EXTERNAL LINKS: ", tpe1 tpe2;   0483
   RETURN;                                                              0484
   END.   %%
```

```
                                                                        02855
(csetlindef)   % GB: core NLS Set Link Default for file procedure %
PROCEDURE (fileno, d1, d2 % => no value %);                             02176
   % Procedure description                                              02177
      FUNCTION                                                          02178
         Every NLS file has a default directory for links; links in the
         file without an explicit directory will use that default.  This
         procedure sets the default directory for the file 'fileno' to the
         directory designated by 'd1' through 'd2'.  It stores the
         directory number in the file header.                           02179
      ARGUMENTS                                                         02180
         fileno - INTEGER - NLS file number for the file                02181
         d1 - TEXT POINTER - start of directory name                    03538
         d2 - TEXT POINTER - end of directory name                      03539
      RESULTS                                                           02182
         none                                                           02183
      NON-STANDARD CONTROL                                              02184
         none                                                           02185
      GLOBALS                                                           02186
         funo, filhed - read only                                       02187
      EXAMPLE                                                           02188
         *s* _ "SMITH";                                                 02189
         FIND SF(*s*) ^tp1 SE(*s*) ^tp2;                                03540
         csetlindef(stid.stfile, $tp1, $tp2);                           03541
      %                                                                 02190
   % Declarations %                                                    02191
      LOCAL slng, fust, funoadr;                                         0415
      REF d1, d2, funoadr, fust;                                         0417
   % make sure we have a locked file %                                   0418
      crepc(fileno);                                                    03690
   &funoadr _ filhdr(fileno) + ($funo - $filhed);                        0426
   funoadr _ 4811 + (IF (slng_d2[1]-d1[1])<20 THEN (fust0-$filhed) ELSE
   (fust1-$filhd1+blksiz)) + fbhdl;                                     03691
   lodhdadd(fileno, funoadr : &fust);                                   03694
   %set string length%                                                  03697
      fust _ IF slng < 20 THEN 24B6 ELSE 112B6;                         03695
   *fust* _ d1 d2, 0;                                                   03696
   fust.L _ fust.L - 1;                                                 03715
   RETURN;                                                               0428
   END.   %%
```

```
                                                                        02194
(csetngro)   % GB: core NLS Set Name delimiter Group procedure %
PROCEDURE (stid1, stid2, dlleft, dlrght, da % => no value %);           02602
   % Procedure description                                             02603
      FUNCTION                                                          02604
         Sets the name delimiters in all statements in the group at 'stid1'
         through 'stid2' to 'dlleft' and 'dlrght'.  Only those statements
         are changed which pass the viewspecs (including content filter, if
         any) in the display area specified.                            02605
      ARGUMENTS                                                         02606
         stid1 - STID - head of group to change                         03542
         stid2 - STID - tail of group to change                         03543
         dldeft - ASCII - new left name delimiter                       03545
         dlrght - ASCII - new right name delimiter                      03546
            ASCII character code, right justified                       03549
         da - DISPLAY AREA - for viewspecs                              03544
      RESULTS                                                           02608
         none                                                           02609
      NON-STANDARD CONTROL                                              02610
         none                                                           02611
      GLOBALS                                                           02612
         none                                                           02613
      EXAMPLE                                                           02614
         csetngro(stid1, stid2, '(, '), lda())                          02615
      %                                                                 02616
   % Declarations %                                                    02617
      REF da;                                                           0432
   %uses the SEQUENCE GENERATOR%                                        02623
     %msntx();%                                                         02624
   stid1 _ grptst (stid1, stid2 :stid2);                                03668
   xnmdlset (stid1, stid2, dlleft, dlrght, &da);                        03669
     %msftx();%                                                         02627
   RETURN;                                                              02628
   END.   %%
```

```
                                                                        02620
(csetnsta)   % GB: core NLS Set Name delimiter Statement procedure %
PROCEDURE (stid, dlleft, dlrght % => no value %);                       02630
   % Procedure description                                              02631
      FUNCTION                                                          02632
         Sets the name delimiters in the statement at 'stid' to 'dlleft'
         and 'dlrght'.                                                  03550
      ARGUMENTS                                                         02634
         stid - STID - statement to change                              03551
         dldeft - ASCII - new left name delimiter                       03553
         dlrght - ASCII - new right name delimiter                      03554
            ASCII character code, right justified                       03555
      RESULTS                                                           02636
         none                                                           02637
      NON-STANDARD CONTROL                                              02638
         none                                                           02639
      GLOBALS                                                           02640
         none                                                           02641
      EXAMPLE                                                           02642
         csetnsta(stid, '(, ') )                                        02643
         csetnsta(stid,  0,  0 )                                        03557
            NULL NULL delimiters                                        03559
      %                                                                 02644
   % Declarations %                                                     02645
     %msntx();%                                                          0442
   nmdisset(stid, dlleft, dlrght);                                       0443
     %msftx();%                                                          0445
   RETURN;                                                               0446
   END.   %%
```

```
                                                                    02648
(csettemmod)   % U8: core NLS Set Temporary Modifications to file %
PROCEDURE (fileno % => no value %);                                 02856
   % Procedure description                                          02857
      FUNCTION                                                      02858
         none                                                       02859
      ARGUMENTS                                                     02860
         none                                                       02861
      RESULTS                                                       02862
         none                                                       02863
      NON-STANDARD CONTROL                                          02864
         none                                                       02865
      GLOBALS                                                       02866
         none                                                       02867
      EXAMPLE                                                       02868
         none                                                       02869
      %                                                             02870
   % Declarations %                                                02871
      LOCAL fl;                                                      0450
      REF fl;                                                        0451
   &fl _ fintadr(fileno);                                            0452
   IF fl.flpart THEN                                                 0453
      err($"You are already modifying this file!");                  0454
   fl.flbrws _ TRUE;                                                 0455
   RETURN;                                                           0456
   END.   %%
```

```
                                                                        02874
(lupper)   % LB: sets first letters of words to upper case %
PROCEDURE (ptr1, ptr2 % => no value %);                                 02875
   % Procedure description                                              02876
      FUNCTION                                                          02877
         Edits the text string between text pointers ptr1 and ptr2, forcing
         the first alpha of each word to upper case.                    02878
      ARGUMENTS                                                         02879
         none                                                           02880
      RESULTS                                                           02881
         none                                                           02882
      NON-STANDARD CONTROL                                              02883
         none                                                           02884
      GLOBALS                                                           02885
         none                                                           02886
      EXAMPLE                                                           02887
         none                                                           02888
      %                                                                 02889
   % Declarations %                                                     02890
      LOCAL i, alphaflag, char;                                          0376
      LOCAL STRING temp[2000];                                           0377
      REF ptr1, ptr2;                                                    0378
   alphaflag _ FALSE;                                                    0379
   *temp* _ ptr1 ptr2; %make local copy of string%                       0380
   FOR i _ 1 UP UNTIL > temp.L DO                                        0381
      CASE char _ *temp*[ i ] OF                                         0382
         IN ['a, 'z]:                                                    0393
            BEGIN                                                        0384
            IF NOT alphaflag THEN *temp*[ i ] _ char - 40B;              0385
            alphaflag _ TRUE;                                            0386
            END;                                                         0387
         IN ['A, 'Z]:                                                    0389
            BEGIN                                                        0390
            IF alphaflag THEN *temp*[ i ] _ char + 40B;                  0391
            alphaflag _ TRUE;                                            0392
            END;                                                         0393
         = '': %no caps after apostrophe inside word%                  03209
            NULL;                                                       03210
         ENDCASE alphaflag _ FALSE;                                      0394
   ST ptr1 ptr2 _ *temp*; % replace edited string %                      0395
   RETURN;                                                               0396
   END.   %%
```

```
                                                                        02893
%show%                                                                   0486
   (csnoarcdir)   % UB: core NLS Show Archive Directory procedure %
   PROCEDURE (d1, d2, astr % => no value %);                            02894
      % Procedure description                                          02895
         FUNCTION                                                       02896
            Not implemented yet.                                        02897
         ARGUMENTS                                                      02898
            none                                                        02899
         RESULTS                                                        02900
            none                                                        02901
         NON-STANDARD CONTROL                                           02902
            none                                                        02903
         GLOBALS                                                        02904
            none                                                        02905
         EXAMPLE                                                        02906
            none                                                        02907
         %                                                              02908
      % Declarations %                                                 02909
         REF astr, d1, d2;                                              0489
      err( notyet );                                                    0490
      RETURN;                                                           0491
      END.   %%
```

```
                                                                        02912
(cshodir)   % GB: core NLS Show Directory procedure %
PROCEDURE (info, gropk, sortk, rhstn, fname % => no value %);           02088
   % Procedure description                                             02089
      FUNCTION                                                          02090
         Prints the names of the files specified by 'fname' into the
         literal display area.                                          02091
      ARGUMENTS                                                         02092
         info - DLINFO - record describing what to list, or 0           03576
         gropk - DLGRP - record containing grouping information, or 0   03577
         sortk - DLSORT - record containing sorting information, or 0   03578
         rhstn - INTEGER -                                              03569
            number of the host on which the files reside.  Only the local
            host ('lhostn') is currently implemented.                   03570
         fname - STRING -                                               03571
            file group name string in TENEX format.  Connected directory is
            assumed if a directory is not specified.  Examples:         03572
               "*.*"                                                    03573
               "<SMITH>*.PC"                                            03574
               "<SMITH>FOO.NLS;3"                                       03575
      RESULTS                                                           02094
         none                                                           02095
      NON-STANDARD CONTROL                                              02096
         Error if                                                       03579
            host is not the local host                                  03580
            files are illegal or not disk files                         03581
            usual TENEX file system errors                              03583
            various internal errors                                     03584
      GLOBALS                                                           02098
         <lots>                                                         02099
      EXAMPLE                                                           02100
         cshodir(0, 0, 0, lhostn, $"*.NLS;*")                          02101
      %                                                                 02102
   % Declarations %                                                    02103
      LOCAL                                                              0501
         noverf,      % no version numbers flag %                        0502
         noextf,      % no extension names flag %                        0503
         sortdi,              % sort direction %                         0504
         tptr,                % temp pointer for building final string %
                                                                         0505
         chns,                % pointer to data structure %              0506
         adlmb,       % address of directory list storage %             0507
         fjfn,                % main jfn %                               0508
         jfnflgs,     % left half flags for a group jfn %               0509
         flags                % bits indicating * typed for file name fields
         %                                                               0510
         ;                                                               0511
      LOCAL TEXT POINTER                                                 0512
         tp1, tp2, tp3;                                                  0513
      LOCAL STRING                                                       0514
         nwname[200],         % current file name link %                 0515
         oldname[200],        % previous file name link %                0516
         astr[2000], % string for listing a file %                       0517
         uname[200], % complete name as entered by user %                0518
         udirname[40],        % user input directory name %              0519
         ufilname[40],        % user input file name %                   0520
```

```
        uextname[40],        % user input extension name %                0521
        uverno[40], % user input version number %                        0522
        ulftovr[40],         % user input beyond version number %         0523
        jfnname[200],        % complete name from jfns %                  0524
        errstring[200],      % eror message string %                      0525
        tstring[200]         % temporary string %                        0526
        ;                                                                 0527
    REF fname, tptr, chns, adlmb;                                         0528
% initial variables %                                                    0530
    fjfn _ jfnflgs _ flags _ &tptr _ &chns _ &adlmb _ 0;                  0531
    sortdi _ sortk.dlsrvr := FALSE;                                       0532
    noverf _ info.dlinvr := FALSE;                                        0533
    noextf _ info.dlinex := FALSE;                                        0534
% find out if remote or local (disk) file %                              0535
    CASE rhstn OF                                                         0536
       = lhostn:  NULL;                                                   0537
       ENDCASE err( $"remote file manipulations not implemented yet" );
                                                                          0538
% now parse user input strings %                                         0539
    parseinput( &fname, $udirname, $ufilname, $uextname, $uverno,
    $ulftovr, $uname, $flags);                                            0540
% get main jfn (old file only, group jfn) %                              0541
    IF info.dlidlt THEN                                                   0542
       BEGIN                                                              0543
       IF NOT                                                             0544
          (fjfn _                                                         0545
             sgtjfn(                                                      0546
                gtjoif .V gtjstr .V gtjidl,                               0547
                $uname, $errstring : jfnflgs                              0548
             )                                                            0549
          )                                                               0550
             THEN err( $errstring );                                      0551
       END                                                                0552
    ELSE                                                                  0553
       IF NOT                                                             0554
          (fjfn _                                                         0555
             sgtjfn(                                                      0556
                gtjoif .V gtjstr, $uname, $errstring : jfnflgs            0557
             )                                                            0558
          )                                                               0559
             THEN err( $errstring );                                      0560
% now find out if we support this type of file %                         0561
    IF NOT chkdev( fjfn ) THEN err( $"only disk files supported" );       0562
% get directory list storage %                                           0563
    IF NOT (&adlmb _ dlgmb(400)) THEN                                     0564
       err($"Internal Storage Problem");                                  0565
% cleanup on any problems %                                              0566
    ON SIGNAL ELSE                                                        0567
       BEGIN                                                              0568
       dlfmb(&adlmb := 0);                                                0569
       IF NOT SKIP rljfn( fjfn ) THEN NULL;                               0570
       END;                                                               0571
% go create the data structure %                                         0572
    IF NOT  (&chns _                                                      0573
       dldriver(info, gropk, sortk, fjfn, jfnflgs, $errstring, &adlmb))
       THEN                                                               0574
```

```
            BEGIN                                                    0575
            IF NOT SKIP !rljfn( fjfn ) THEN NULL;                    0576
            dlfmb( &adlmb := 0);                                     0577
            IF errstring.L THEN err($errstring)                      0578
            ELSE err($"System Screwup");                             0579
            END;                                                     0580
    % broadcast any error message %                                  0581
      IF errstring.L THEN dismes(2,$errstring);                      0582
    % get ready to list the directory %                             0583
      xdlpnt( typenulllit, $"");                                     0584
    % build string for one file, list it,  and loop for all files %  0585
      WHILE &chns DO                                                 0586
         BEGIN                                                       0587
         IF inptrf THEN EXIT LOOP;                                   0588
         IF sortdi THEN &tptr _ chns.dlgpnu                          0589
         ELSE &tptr _ chns.dlgbsc;                                   0590
         WHILE &tptr DO                                              0591
            BEGIN                                                    0592
            IF inptrf THEN EXIT LOOP 2;                              0593
            IF NOT tptr.dlfbfl.dlstig THEN                           0594
               BEGIN                                                 0595
               astr.L _ 0;                                           0596
               IF noverf OR noextf THEN                              0597
                  BEGIN                                              0598
                  FIND                                               0599
                     SE(*[tptr.dlfbal]*) [",] ^tp3 [";  / ".] ^tp2 [".]
                     ^tp1;                                           0600
                  IF noverf AND noextf THEN                          0601
                     *nwname* _ SF(*[tptr.dlfbal]*) tp1,             0602
                        tp3 SE(*[tptr.dlfbal]*)                      0603
                  ELSE                                               0604
                     IF noverf THEN                                  0605
                        *nwname* _ SF(*[tptr.dlfbal]*) tp2,          0606
                           tp3 SE(*[tptr.dlfbal]*)                   0607
                     ELSE                                            0608
                        *nwname* _ SF(*[tptr.dlfbal]*) tp1,          0609
                           tp2 SE(*[tptr.dlfbal]*);                  0610
                  IF *nwname* = *oldname* THEN GOTO sbypass          0611
                  ELSE *oldname* _ *nwname*;                         0612
                  *astr* _ *astr*, *nwname*;                         0613
                  END                                                0614
               ELSE                                                  0615
                  *astr* _ *astr*, *[tptr.dlfbal]*;                  0616
               % build additional line 1 information string %        0617
                  dlbldai(                                           0618
                     fjfn, tptr.dlfbpf, tptr.dlfbff, tptr.dlfbpf, $astr);
                                                                     0619
               *astr* _ *astr*, CR, LF;                              0620
               % build information string for this file %           0621
                  dlbldi(                                            0622
                     info, tptr.dlfbpf, tptr.dlfbff, tptr.dlfbpf, $"   ",
                     $astr);                                         0623
               % now list this file %                               0624
                  xdlpnt( fbaddlit, $astr );                         0625
               END;                                                  0626
            (sbypass):                                               0627
```

```
            IF sortdi THEN &tptr _ tptr.dlfbpb                    0628
            ELSE &tptr _ tptr.dlfbnb;                             0629
            END;                                                  0630
          &chns _ chns.dlgbnb;                                    0631
          END;                                                    0632
    % get rid of any lingering storage %                          0633
      dlfmb( &adlmb := 0 );                                       0634
    % make sure we get rid of any jfns %                          0635
      IF NOT SKIP !rljfn( fjfn ) THEN NULL;                       0636
    % all done, tell user %                                       0637
      xdlpnt( typecalit, $"" );                                   0638
    % return %                                                    0639
      RETURN;                                                     0640
    END.   %%
```

```
                                                                    02106
(cshodskspa)   % OB: core NLS Show Disk Space status procedure %
PROCEDURE (astr % => integer %);                                    02069
   % Procedure description                                         02070
      FUNCTION                                                     02071
         Computes the amount of disk space left.                   02072
      ARGUMENTS                                                    02073
         astr - STRING - string to receive message                 02074
      RESULTS                                                      02075
         procedure-value - INTEGER - disk space left               02076
      NON-STANDARD CONTROL                                         02077
         none                                                      02078
      GLOBALS                                                      02079
         none                                                      02080
      EXAMPLE                                                      02081
         none                                                      02082
      %                                                            02083
   % Declarations %                                               02084
      LOCAL                                                         0646
         fjfn,          % main jfn %                              0647
         jfnflgs,       % left half flags for a group jfn %       0648
         size,          % size (in pages) of an individual file % 0649
         usrinu,        % in use pages this directory %           0650
         usrall,        % allowed pages this directory %          0651
         usrund,        % undeleted pages this directory %        0652
         usrdel,        % deleted pages this directory %          0653
         sysinu,        % used pages in system %                  0654
         sysfre;        % free pages in system %                  0655
      LOCAL STRING                                                  0657
         dirname[40],           % string to get directory name %  0658
         errstring[200];        % eror message string %           0659
      REF astr;                                                     0661
   % initial variables %                                          0663
      fjfn _ jfnflgs _ usrinu _ usrall _ usrund _ usrdel _ sysinu _ sysfre
      _ 0;                                                        0664
   % get main jfn (old file only, group jfn) %                    0665
      IF NOT (fjfn _ sgtjfn( gtjoif .V gtjstr .V gtjidl, $dirsfname,
      $errstring : jfnflgs ) )THEN                                0666
         BEGIN                                                    0667
         IF (r1.RH = gjfx20) OR (r1.RH = gjfx32) THEN NULL        0668
         ELSE err( $errstring );                                  0669
         END;                                                     0670
   % loop to count usage for this directory %                     0671
      IF fjfn THEN LOOP                                         0672
         BEGIN                                                  0673
         % get number of pages this file %                      0674
            !gtfdb( fjfn, 1000011B, $r3);                       0675
            size _ r3.RH;                                       0676
         % add to total in use count %                          0677
            usrinu _ usrinu + size;                             0678
         % get deletion status of this file %                   0679
            !gtfdb( fjfn, 1000001B, $r3);                       0680
         % now update the appropriate sum %                     0681
            IF r3.fdbdel THEN usrdel _ usrdel + size            0682
            ELSE usrund _ usrund + size;                        0683
         % now get the next file in the group %                 0684
```

```
            (gndsjfn);                                                  0685
            r1.LH _ jfnflgs;                                            0686
            r1.RH _ fjfn;                                               0687
            IF NOT SKIP !gnjfn( r1 ) THEN EXIT LOOP;                    0688
          END;                                                          0689
    % now get rid of any lingering jfns %                               0690
        IF NOT SKIP !rljfn( fjfn ) THEN NULL;                           0691
    % get total allowed for this directory %                            0692
       !gtdal( IF tops20flag THEN -1 ELSE 0 );                          0693
       usrall _ r1;                                                     0694
    % get system totals %                                               0695
       !gdskc( 600000777777B );                                         0696
       sysinu _ r1;                                                     0697
       sysfre _ r2;                                                     0698
    % get connected directory name %                                    0699
       !gjinf();                                                        0700
       gdname( r2.RH, $dirname );                                       0701
    % now build the message string %                                    0702
       *astr* _ "Connected to ", *dirname*, CR, LF, STRING(usrinu), " Total
       pages in use -- ", STRING(usrall), " Allowed, ", STRING(usrund), "
       Undeleted, ", STRING(usrdel), " Deleted", CR, LF, "System Total: ",
       STRING(sysfre), " Pages left, ", STRING(sysinu), " Used";        0703
    % all done now, so return %                                        0704
       RETURN(usrinu - usrall);                                         0705
    END.   %%
```

```
                                                                    02087
(cshofilsta)   % JB: core NLS Show File Status procedure %
PROCEDURE (type, fileno, astrng % => no value %);                   02195
   % Procedure description                                          02196
      FUNCTION                                                      02197
         none                                                       02198
      ARGUMENTS                                                     02199
         type - INTEGER - ALL, link, modifications, size            02200
         fileno - INTEGER -                                         02214
         astrng - STRING -                                          02215
      RESULTS                                                       02201
         none                                                       02202
      NON-STANDARD CONTROL                                          02203
         none                                                       02204
      GLOBALS                                                       02205
         none                                                       02206
      EXAMPLE                                                       02207
         none                                                       02208
      %                                                             02209
   % Declarations %                                                 02210
      REF astrng;                                                    0709
   <AUXCOD, fstatus>(fileno, &astrng, type);                         0711
   RETURN;                                                           0712
   END.   %%
```

```
                                                                    02213
(cshomarfil)   % JB: core NLS Show Markers in File procedure %
PROCEDURE (fileno, astr % => no value %);                           02649
   % Procedure description                                          02650
      FUNCTION                                                      02651
         none                                                       02652
      ARGUMENTS                                                     02653
         none                                                       02654
      RESULTS                                                       02655
         none                                                       02656
      NON-STANDARD CONTROL                                          02657
         none                                                       02658
      GLOBALS                                                       02659
         none                                                       02660
      EXAMPLE                                                       02661
         none                                                       02662
      %                                                             02663
   % Declarations %                                                02664
      REF astr;                                                      0782
   <AUXCOD, seemkr>(fileno, &astr);                                  0783
   RETURN;                                                           0784
   END.   %%
```

```
                                                                    02667
(cshonsta)  % UB: core NLS Show Name delimiters for Statement %
PROCEDURE (stid, astrng % => no value %);                           02573
   % Procedure description                                          02574
      FUNCTION                                                      02575
         none                                                       02576
      ARGUMENTS                                                     02577
         none                                                       02578
      RESULTS                                                       02579
         none                                                       02580
      NON-STANDARD CONTROL                                          02581
         none                                                       02582
      GLOBALS                                                       02583
         none                                                       02584
      EXAMPLE                                                       02585
         none                                                       02586
      %                                                             02587
   % Declarations %                                                02588
      LOCAL dlleft, dlrght;                                         02593
      REF astrng;                                                   02594
   dlleft _ getnmdl( stid : dlrght);                                 0790
   IF dlleft = 0 THEN *astrng* _ "NULL", SP                          0791
   ELSE *astrng* _ dlleft, SP;                                       0792
   IF dlrght = 0 THEN *astrng* _ *astrng*, "NULL"                    0793
   ELSE *astrng* _ *astrng*, dlrght;                                 0794
   RETURN;                                                          02590
   END.   %%
```

```
                                                                        02591
(csnofrring)   % JB: core NLS Show status of File Return Ring %
PROCEDURE (da, astrng % => no value %);                                 02913
   % Procedure description                                              02914
      FUNCTION                                                          02915
         none                                                           02916
      ARGUMENTS                                                         02917
         none                                                           02918
      RESULTS                                                           02919
         none                                                           02920
      NON-STANDARD CONTROL                                              02921
         none                                                           02922
      GLOBALS                                                           02923
         none                                                           02924
      EXAMPLE                                                           02925
         none                                                           02926
      %                                                                 02927
   % Declarations %                                                     02928
      LOCAL end, i, jfn, filstr, fileno, flags;                          0716
      LOCAL TEXT POINTER t1, t2, u1, u2, f1, f2, n1, n2, v1, v2;         0717
      LOCAL STRING locstr[200];                                          0718
      REF da, astrng;                                                    0719
   *astrng* _ NULL;                                                      0720
   IF NOT da.daexis THEN                                                 0721
      err($"Illegal display area address in cshofrring");                0722
   end _ frrlength(da.dalink);                                           0723
   FOR i _ 0 UP UNTIL > end DO                                           0724
      BEGIN                                                              0725
      ON SIGNAL ELSE REPEAT LOOP;  %in case top entry non-existant%     03672
      filstr _ readfrring(da.dalink, i);                                 0726
      ON SIGNAL ELSE;                                                   03673
      FIND SF(*[filstr]*) ^t1;                                           0727
      lnbfls( $t1, 0, $locstr);                                          0728
      IF ( fileno _ filnum($locstr) ) = -1 THEN                          0729
         BEGIN        %file not open -- get a jfn%                       0730
         flags _ getgtjflg(write, origff, oldvrsn);                     0731
         jfn _ sgtjfn(flags, $locstr, $locstr);                         0732
         END                                                            0733
       ELSE jfn _ [flntadr(fileno)].florig;                             0734
      r3 _ 0;                                                           0735
      IF jfn THEN                                                       0736
         BEGIN                                                          0737
         *locstr* _ NULL;                                               0738
         !gtfdb(jfn, 1000024B, $r3);                                    0739
         END;                                                           0740
      IF r3.lkinit THEN                                                 0741
         lockid($locstr, r3.lkdirn, r3.lkinit);                         0742
      *astrng* _ *astrng*, *[filstr]*, *locstr*, CR, LF;                0743
      IF fileno = -1 THEN reljfn(jfn);                                  0744
      END;                                                              0745
   RETURN;                                                              0746
   END.   %%
```

```
                                                                    02931
(cshosrring)   % UB: core NLS Show status of Statement Return Ring %
PROCEDURE (da, astr % => no value %);                               02951
   % Procedure description                                          02952
      FUNCTION                                                      02953
         none                                                       02954
      ARGUMENTS                                                     02955
         none                                                       02956
      RESULTS                                                       02957
         none                                                       02958
      NON-STANDARD CONTROL                                          02959
         none                                                       02960
      GLOBALS                                                       02961
         none                                                       02962
      EXAMPLE                                                       02963
         none                                                       02964
      %                                                             02965
   % Declarations %                                                02966
      LOCAL stid, stdb, len, cc, pvs1, pvs2, srr, i;                 0751
      LOCAL TEXT POINTER tp1,tp2;                                    0752
      LOCAL STRING temp[70];                                         0753
      REF da, srr, astr;                                             0754
   % advance through jump stack %                                   0756
      readfrring(da.dalink, 0 : &srr);                               0757
      srrcnt _ srrlength( &srr );                                    0758
   FOR i _ 1 UP UNTIL >= srrcnt DO                                   0759
      BEGIN                                                          0760
      stid _ readsrring(&srr, i : cc, pvs1, pvs2);                   0761
      %display the current statement%                                0762
         % display first n chars of stmt in name area %              0763
            % get statement length %                                 0764
               IF NOT lodprop( stid, txttyp : stdb) THEN            0765
                  err($"No text block associated with node");       0766
               len _ [stdb].schars + 1; % number of chars %         0767
            % construct text ptrs to each end of stmt %             0768
               tp1 _ stid;                                          0769
               tp1[1] _ 1;                                          0770
               tp2 _ stid;                                          0771
               %truncate if over n chars %                          0772
                  tp2[1] _ MIN(IF nlmode = fulldisplay THEN 60 ELSE
                  20,len);                                          0773
            % assign something to the string "temp" %              0774
               IF len > 1 THEN *temp* _ tp1 tp2 ELSE *temp* _ "<NULL>";
                                                                    0775
      *astr* _ *astr*, *temp*, CR, LF;                              0776
      END;                                                          0777
   RETURN;                                                          0778
   END.   %%
```

```
                                                                        02969
%sort%                                                                   0797
   (csorgro)   % GB: core NLS Sort Group procedure %
   PROCEDURE (bug1, bug2 % => no value %);                              02381
      % Procedure description                                          02382
         FUNCTION                                                       02383
            Sorts the statements in the group at 'bug1' through 'bug2'
            according to the currently instituted sort key, if any.  If there
            isn't one, the default of sorting alphabetically without regard to
            case is used.  The sort key program is taken from the field
            'daukeycod' in the current display area.                    02384
         ARGUMENTS                                                      02385
            bug1 - TEXT POINTER - head of group to sort                 02386
            bug2 - TEXT POINTER - tail of group to sort                 03585
         RESULTS                                                        02387
            none                                                        02388
         NON-STANDARD CONTROL                                           02389
            none                                                        02390
         GLOBALS                                                        02391
            none                                                        02392
         EXAMPLE                                                        02393
            LOCAL da;  REF da;                                          02394
            &da _ lda();                                                03586
            da.daukeycod _ $procedure;                                  03587
            csorgro($tp1, $tp2);                                        03588
         %                                                              02395
      % Declarations %                                                  02396
         REF bug1,bug2;                                                  0800
      <CULSRT, sortgrp> (bug1, bug2);                                    0801
      RETURN;                                                           02398
      END.   %%
```

```
                                                                        02399
(csort)   % LR: core NLS Sort primitive %
PROCEDURE (bugx, bug1, bug2, rlevcnt, kpr, copyf % => no value %);      02970
   % Procedure description                                              02971
      FUNCTION                                                          02972
         none                                                           02973
      ARGUMENTS                                                         02974
         none                                                           02975
      RESULTS                                                           02976
         procedure-value                                                02977
      NON-STANDARD CONTROL                                              02978
         none                                                           02979
      GLOBALS                                                           02980
         none                                                           02981
      EXAMPLE                                                           02982
         none                                                           02983
      %                                                                 02984
   % Declarations %                                                     02985
   <COLSRT, sort> (bugx,bug1,bug2,rlevcnt,kpr,copyf);                    0805
   RETURN;                                                              02987
   END.   %%
```

```
                                                                        02988
%substitute%                                                             0807
   (csubgro)   % UB: core NLS Substitute Group procedure %
   PROCEDURE (stid1, stid2, sbheadb, da % => no value %);                02497
      % Procedure description                                           02498
         FUNCTION                                                       02499
            none                                                        02500
         ARGUMENTS                                                      02501
            none                                                        02502
         RESULTS                                                        02503
            none                                                        02504
         NON-STANDARD CONTROL                                           02505
            none                                                        02506
         GLOBALS                                                        02507
            none                                                        02508
         EXAMPLE                                                        02509
            none                                                        02510
         %                                                              02511
      % Declarations %                                                  02512
         LOCAL stid, vspec,                                             0810
            sw;   % address of sequence work area %                     0811
         REF da, sbheadb, sw;                                           0812
      IF &da NOT IN [$dpyarea, $dpyarea + dal*dacnt) OR                 0814
         NOT da.daexis OR da.daempty THEN err($"Illegal Substitute");   0815
        %msntx();%                                                      0816
      dismes(1, $"Substitute In Progress");                             0817
      stid1 _ grptst(stid1, stid2 : stid2);                             0818
      vspec _ da.davspec;                                               0819
      vspec.vsbrof _ vspec.vsplxf _ FALSE;                              0820
      &sw _ openseq (stid1, stid2, vspec, da.davspc2, da.dausqcod,
      da.dacacode);                                                     0821
      WHILE (stid _ seqgen (&sw)) # endfil DO csubsta(stid, &sbheadb, &da);
                                                                        0822
      closeseq (&sw);                                                   0823
      dismes (0);                                                       0824
        %msftx();%                                                      0825
      RETURN;                                                           0826
      END.   %%
```

```
                                                                        02515
(csubsta)   % UR: core NLS Substitute Statement procedure %
PROCEDURE (stid, sbheadb, da % => no value %);                          02516
   % Procedure description                                             02517
      FUNCTION                                                          02518
         none                                                           02519
      ARGUMENTS                                                         02520
         none                                                           02521
      RESULTS                                                           02522
         none                                                           02523
      NON-STANDARD CONTROL                                              02524
         none                                                           02525
      GLOBALS                                                           02526
         none                                                           02527
      EXAMPLE                                                           02528
         none                                                           02529
      %                                                                 02530
   % Declarations %                                                     02531
      LOCAL                                                              0831
         cary,   %pointer (later SKIPN) to subdsp%                       0832
         ttype,  %type of text entity being substituted%                 0833
         sfc,    %first char position for scan%                          0834
         sbp,    %byte pointer to chars in statement%                    0835
         sfbp,   %initial byte pointer in statement%                     0836
         lbp,    %byte pointer to left delimiter%                        0837
         rbp,    %byte pointer to right delimiter%                       0838
         snc,    %counter for chars in statement%                        0839
         ch,     %last char read from statement%                         0840
         chbp,   %byte ptr to end of last change%                        0841
         chnc,   %char pos of last change from right end of original
         statement%                                                      0842
         chflag, %count of changes made%                                 0843
         cap,    %ptr to substitution description record%               0844
         cbp,    %byte ptr to candidate or replacement%                  0845
         cnc,    %length of candidate or replacement%                    0846
         cnc1,   %length of candidate%                                   0847
         tbp,    %temp sbp for comparison%                               0848
         wbp,    %byte ptr to last char written in new statement%        0849
         wnc,    %number of chars written in new statement%              0850
         maxsi;  %max size of new statement%                             0851
      POINTER sbheadb, da, cap;                                          0852
     %msntx();%                                                          0853
   cary _ sbheadb.sbdp;                                                  0854
   ttype _ sbheadb.sbtt;                                                 0855
   sfc _ IF stid.stastr OR da.davspec.vsnamf THEN 1 ELSE fchtxt(stid);   0856
   sfbp _ sbp _ stbpget(stid, sfc: snc);                                 0857
      snc _ snc + 1 - sfc;                                               0858
   IF sfc = 1 THEN BEGIN %entire statement%                              0859
      chbp _ sbp; chnc _ snc;                                            0860
   END ELSE %name not scanned%                                           0861
      chbp _ stbpget(stid, 1: chnc);                                     0862
   chflag _ 0;                                                           0863
   maxsi _ lit.M;                                                        0864
   wbp _ 010788 + $lit; %text begins at $lit+1%                          0865
   wnc _ 0;                                                              0866
   % set up for fast scan %                                             0867
```

```
      a1 _ skipni;                                                    0868
      !HLLM a1,cary; %insert instruction part%                        0869
   (step1):                                                           0870
      BUMP snc;                                                       0871
   (step): %fast scan loop%                                           0872
      !SOSG snc;                                                      0873
      !JRST done;                                                     0874
      !ILDB a1,sbp;                                                   0875
      !XCT cary; %SKIPN a2,subdsp(a1)%                                0876
      !JRST step;                                                     0877
      cap _ a2;                                                       0878
      DO                                                              0879
         BEGIN                                                        0880
         IF (cnc _ (cnc1 _ cap.catnc)-1) < snc THEN                   0881
            BEGIN                                                     0882
            tbp _ sbp;                                                0883
            cbp _ cap.catbp;                                          0884
            ch _ ^cbp; %skip first char%                              0885
            FOR cnc DOWN UNTIL <=0 DO                                 0886
               IF ^tbp # ^cbp THEN GOTO noteq;                        0887
            % found match, check delimiters %                         0888
            IF (lbp _ bkbp(sbp)) # sfbp THEN %check left delimiter%   0889
               IF NOT sbdelck(.lbp,ttype) THEN GOTO noteq;            0890
            IF snc # cnc1 THEN %check right delimiter%                0891
               BEGIN                                                  0892
               rbp _ tbp;                                             0893
               IF NOT sbdelck(^rbp,ttype) THEN GOTO noteq;            0894
               END;                                                   0895
            IF (wnc _ wnc + chnc - snc + (cnc _ cap.carnc))           0896
               > maxsl THEN                                           0897
                  BEGIN                                               0898
                  % generate string system overflow error %           0899
                  lit.L _ lit.M;                                      0900
                  *lit* _ *lit*, SP;                                  0901
                  RETURN; %if no trap - may not be possible%          0902
                  END;                                                0903
            % copy segment between last match and this %               0904
            UNTIL chbp = lbp DO                                       0905
               ^wbp _ ^chbp;                                          0906
            % enter replacement %                                     0907
            cbp _ cap.carbp;                                          0908
            FOR cnc DOWN UNTIL <=0 DO                                 0909
               ^wbp _ ^cbp;                                           0910
            % update variables %                                      0911
            chbp _ sbp _ tbp;                                         0912
            chnc _ snc _ snc - cnc1;                                  0913
            BUMP chflag;                                              0914
            GOTO step1;                                               0915
         END;                                                         0916
         (noteq):                                                     0917
      END WHILE cap _ cap.canxt;                                      0918
      GOTO step;                                                      0919
   (skipni): !SKIPN a2,0(a1); %instruction part for cary%             0920
   (done):                                                            0921
   IF chflag THEN BEGIN                                               0922
      IF (wnc _ wnc + chnc) > maxsl THEN                              0923
```

```
        dismes(2, $"New statement is too long -- substitution not made")
                                                                        0924
    ELSE                                                                0925
        BEGIN                                                           0926
        % copy rest of statement (past last match) %                    0927
          UNTIL chbp = sbp DO                                           0928
            ^wbp _ ^chbp;                                               0929
        lit.L _ wnc;                                                    0930
        FIND SF(stid) ^sptr1 SE(stid) ^sptr2; %remove markers%          0931
        cldsr($sptr1);                                                  0932
        ST stid _ *lit*;                                                0933
        subcnt _ subcnt + chflag;                                       0934
        END;                                                            0935
END;                                                                    0936
  %msftx();%                                                            0937
RETURN;                                                                 0938
END.   %%
```

```
                                                                        02534
% Low-level substitute routines %                                      03148
   (bkbp) PROCEDURE (bp); %back up byte pointer%                         0951
      IF (bp _ bp + 07B10) < 0 THEN                                      0952
         bp _ bp - 430000000001B;                                        0953
      RETURN(bp) END.                                                   0954
                                                                        0955
   (sbdelck) PROCEDURE (ch, ttype); %check delimiter%                   0956
      CASE ttype OF                                                     0957
         =numbrv: IF ch=0 OR ch=". OR ch=', THEN RETURN(FALSE);         0958
         =wordv: IF ch=LD THEN RETURN(FALSE);                           0959
         =visv: IF ch=PT THEN RETURN(FALSE);                            0960
         =invisv: IF ch=NP THEN RETURN(FALSE);                          0961
         ENDCASE;                                                       0962
      RETURN(TRUE) END.                                                 0963
                                                                        0964
   (stopget) PROCEDURE (stid, cnt); %statement byte pointer get%        0941
      LOCAL bp;                                                         0942
      swork _ stid; swork[1] _ cnt;                                     0943
      fechcl(1, $swork);                                                0944
      %swork[2] contains byte count+1, swork[4] has byte pointer%       0945
      bp _ swork[4];                                                    0946
      IF bp .A 77B10 = 44B10 THEN                                       0947
         bp _ bp - 430000000001B;                                       0948
         %adjust char and word pos%                                     0949
      RETURN(bp, swork[2]-1) END.   %%
```

```
                                                                        0950
%transpose%                                                             0965
   (ctragro)   % GR: core NLS Transpose Group procedure %
   PROCEDURE (first1, first2, secnd1, secnd2, filter, vsptr % => statement,
   statement, statement, statement %);                                  02301
      % Procedure description                                          02302
        FUNCTION                                                        02303
           Transposes the group at 'first1' through 'first2' (first group)
           with the group at 'secnd1' through 'secnd2' (second group).  The
           first group need not be ahead of the second in the file structure.
           Assumes that the groups are legal, but checks to see that the
           groups don't overlap.                                        02320
           If one group follows the other immediately, then the second group
           is merely deleted from the structure and inserted after the pred
           (or source) of the first; the first is in the right position
           automatically.  Otherwise, the routine finds the pred (or source)
           of each group.                                               02321
           If the two groups are in the same file, it calls REMGRP to delete
           one group from the structure, then INSGRP to insert the group
           after the pred (or source) of the other group.  The same
           delete/insert sequence is followed for the other group.
           Otherwise, it uses COPGROUP twice and XDELE twice to perform the
           requisite operations.                                        02322
        ARGUMENTS                                                       02305
           first1 - STID - head of first group to transpose             03596
           first2 - STID - tail of first group to transpose             03597
           secnd1 - STID - head of second group to transpose            03603
           secnd2 - STID - tail of second group to transpose            03604
           filter - BOOLEAN -                                           03598
              TRUE  => filter statements                                03599
              FALSE => transpose all statements                         03600
           vsptr - VIEWSPECS -                                          03601
              viewspecs through which to filter statements, or 0.       03602
        RESULTS                                                         02307
           procedure-value - STID - head of second group transposed     03607
           2nd-result - STID - tail of second group transposed          03608
           3rd-result - STID - head of first group transposed           03605
           4th-result - STID - tail of first group transposed           03606
              i.e. the stids in transposed order                        03622
        NON-STANDARD CONTROL                                            02309
           Error if a group can't be moved, e.g. no write access to file
                                                                        02310
        GLOBALS                                                         02311
           levdown, levsuc, endfil - read only                          02312
        EXAMPLE                                                         02313
           stid3 _                                                      02314
              ctragro(stid1, stid2, stid3, stid4, FALSE, 0 : stid4, stid1,
              stid2)                                                    03609
        %                                                               02315
      % Declarations %                                                  02316
        LOCAL                                                            0972
           work,   %temp%                                                0973
           f1, f2, s1, s2,     %temps for return stids%                  03667
           gp1tgt, %target for group delimited by first1,first2%         0974
           gp1dir, %direction for insertion of above group%              0975
           gp2tgt, %target for group delimited by secnd1,secnd2%         0976
```

```
        gp2dir; %direction for insertion of above group%                    0977
      REF vsptr;                                                            0971
     %msnst();%                                                            03334
  first1 _ grptst(first1, first2 : first2);                                 03335
  secnd1 _ grptst(secnd1, secnd2 : secnd2);                                 03336
  trntst(first1, first2, secnd1, secnd2);                                   03337
  IF getfhd(secnd1) THEN                                                    03338
     BEGIN                                                                  03339
     gp1tgt _ getup(secnd2);                                                03340
     gp1dir _ levdown;                                                      03341
     END                                                                    03342
  ELSE                                                                      03343
     BEGIN                                                                  03344
     gp1dir _ levsuc;                                                       03345
     IF (gp1tgt _ getprd(secnd1)) = first2 THEN                             03346
        IF NOT filter THEN                                                  03347
           BEGIN %SECND1,2 follows FIRST1,2 directly%                       03348
           IF NOT remgrp(first1, first2) THEN                               03349
              err($"illegal transpose");                                    03350
           insgrp(secnd2, levsuc, first1, first2);                          03351
           RETURN;                                                          03352
           END;                                                             03353
     END;                                                                   03354
  IF getfhd(first1) THEN                                                    03355
     BEGIN                                                                  03356
     gp2tgt _ getup(first2);                                                03357
     gp2dir _ levdown;                                                      03358
     END                                                                    03359
  ELSE                                                                      03360
     BEGIN                                                                  03361
     gp2dir _ levsuc;                                                       03362
     IF (gp2tgt _ getprd(first1)) = secnd2 THEN                             03363
        IF NOT filter THEN                                                  03364
           BEGIN %FIRST1,2 follows SECND1,2 directly%                       03365
           IF NOT remgrp(secnd1, secnd2) THEN                               03366
              err($"illegal transpose");                                    03367
           insgrp(first2, gp2dir, secnd1, secnd2);                          03368
           RETURN;                                                          03369
           END;                                                             03370
     END;                                                                   03371
  IF first1.stfile = secnd1.stfile THEN                                     03372
     IF NOT filter THEN                                                     03373
        BEGIN %intrafile transpose%                                         03374
        IF NOT remgrp(secnd1, secnd2) THEN                                  03375
           err($"illegal transpose");                                       03376
        insgrp(gp2tgt, gp2dir, secnd1, secnd2);                             03377
        IF NOT remgrp(first1, first2) THEN                                  03378
           err($"illegal transpose");                                       03379
        insgrp(gp1tgt, gp1dir, first1, first2);                             03380
        END                                                                 03381
     ELSE                                                                   03382
        BEGIN %intrafile filtered transpose%                                03383
        IF gp1tgt = first2 THEN                                             03384
           BEGIN                                                            03385
           IF getfhd(first1) THEN                                           03386
              BEGIN                                                         03387
```

```
            gp1tgt _ getup(first1);                                          03388
            gp1dir _ levdown;                                                03389
            END)                                                             03390
         ELSE                                                                03391
            BEGIN                                                            03392
            gp1dir _ levsuc;                                                 03393
            gp1tgt _ getprd(first1);                                         03394
            END;                                                             03395
         END;                                                                03396
      mvdlfgrp(                                                              03397
         first1, first2, &vsptr, IF (work _ getnxt(getend(first2))) #
         endfil THEN work ELSE getbck(first1), trnsflag, gp1tgt,
         gp1dir);                                                            03398
      IF (gp2tgt = first2) OR (gp2tgt = secnd2) THEN                         03399
         BEGIN                                                               03400
         IF getfhd(secnd1) THEN                                              03401
            BEGIN                                                            03402
            gp2tgt _ getup(secnd1);                                          03403
            gp2dir _ levdown;                                                03404
            END                                                              03405
         ELSE                                                                03406
            BEGIN                                                            03407
            gp2dir _ levsuc;                                                 03408
            gp2tgt _ getprd(secnd1);                                         03409
            END;                                                             03410
         END;                                                                03411
      mvdlfgrp(                                                              03412
         secnd1, secnd2, &vsptr, IF (work _ getnxt(getend(secnd2))) #
         endfil THEN work ELSE getbck(secnd1), trnsflag, gp2tgt,
         gp2dir);                                                            03413
      END                                                                    03414
   ELSE                                                                      03415
      BEGIN %cross file transpose%                                           03416
      IF NOT filter THEN                                                     03417
         BEGIN                                                               03418
         f1 _ copgrp(gp1tgt, gp1dir, first1, first2, TRUE: f2);              03419
         s1 _ copgrp(gp2tgt, gp2dir, secnd1, secnd2, TRUE: s2);              03420
         END                                                                 03421
      ELSE                                                                   03422
         BEGIN                                                               03423
         f1 _ copfgrp(gp1tgt, gp1dir, first1, first2, &vsptr: f2);           03424
         s1 _ copfgrp(gp2tgt, gp2dir, secnd1, secnd2, &vsptr: s2);           03425
         END;                                                                03426
      cdelgro(first1, first2, FALSE, 0);                                     03427
      cdelgro(secnd1, secnd2, FALSE, 0);                                     03428
      secnd1 _ s1;  secnd2 _ s2;                                             03429
      first1 _ f1;  first2 _ f2;                                             03430
      END;                                                                   03431
     %msfst();%                                                              03432
   RETURN(secnd1, secnd2, first1, first2);                                   03433
   END.                                                                      03434

(ctrasta)   % GB: core NLS Transpose Statement procedure %
PROCEDURE (stid1, stid2, filter, vsptr % => statement, statement %);  02323
   % Procedure description                                                  02324
      FUNCTION                                                             02325
```

```
      Interchanges the statements at 'stid1' and 'stid2'.  If 'filter'
      is TRUE, transposes the statements only if they pass the viewspecs
      in 'vsptr'.                                                    03610
   ARGUMENTS                                                         02327
      stid1 - STID - first statement to transpose                    03611
      stid2 - STID - second statement to transpose                   03612
      filter - BOOLEAN -                                             03615
         TRUE  => filter statements                                  03616
         FALSE => transpose all statements                           03617
      vsptr - VIEWSPECS -                                            03618
         viewspecs through which to filter statements, or 0.         03619
   RESULTS                                                           02329
      procedure-value - STID - second statement transposed           02330
      2nd-result - STID - first statement transposed                 03620
         i.e. the stids in transposed order                          03621
   NON-STANDARD CONTROL                                              02331
      Error if one of the stids is an origin statement or an illegal
      stid                                                           02332
   GLOBALS                                                           02333
      none                                                           02334
   EXAMPLE                                                           02335
      stid2 _ ctrasta(stid1, stid2, FALSE, 0 : stid1)                02336
   %                                                                 02337
% Declarations %                                                     02338
   LOCAL                                                             01080
      % variables for intra-file transpose %                         01081
      rngloc1,  % ring location of stid1 %                           01082
      rngloc2,  % ring location of stid2 %                           01083
      sub1,     % psids of sub for stid1 %                           01084
      suc1,     % psids of suc for stid1 %                           01085
      hedf1,                                                         01086
      tailf1,                                                        01087
      up1,      % stid (!) of up of stid1 %                         01088
      pred1,    % stid (!) of pred of stid1 %                       01089
      tailsub1, % stid of tail of sub of 1; zero if 1 has no sub %   01090
      sub2,     % psids of sub for stid2 %                           01091
      suc2,     % psids of suc for stid2 %                           01092
      hedf2,                                                         01093
      tailf2,                                                        01094
      up2,      % stid (!) of up of stid2 %                         01095
      pred2,    % stid (!) of pred of stid2 %                       01096
      tailsub2, % stid of tail of sub of 2; zero if 2 has no sub %   01097
      % variables for inter-file transpose %                        01098
      newsd1,   %new stid, replaces stid1 in cross-file              01099
      trpose%                                                        01100
      newsd2,   %new stid, replaces stid2 in cross-file              01101
      trpose%                                                        01102
      tmpstd;   %temporary to save stid%                             01103
   LOCAL TEXT POINTER t1, t2;                                        01104
   REF vsptr;                                                        01079
  %msnst();%                                                         03233
IF NOT filter THEN                                                   03234
   BEGIN                                                             03235
   FIND SF(stid1) ^t1 SE(stid1) ^t2;                                 03236
   <TXTEDT, cldtxt>($t1, $t2);                                       03237
   FIND SF(stid2) ^t1 SE(stid2) ^t2;                                 03238
```

```
<TXTEDT, cldtxt>($t1, $t2);                                          03239
IF stid1.stfile = stid2.stfile THEN                                  03240
   BEGIN  %intrafile transpose%                                      03241
   IF stid1.stpsid = origin OR stid2.stpsid = origin THEN            03242
      err($"Illegal transpose");                                     03243
   % Save stid1 ring values and up and predecessor %                 03244
      lodent( stid1, rngtyp : rngloc1 );                             03245
      IF [rngloc1].rsid = 0 THEN err($"Bad statement identifier");
                                                                     03246
      sub1 _ [rngloc1].rsub;                                         03247
      suc1 _ [rngloc1].rsuc;                                         03248
      hedf1 _ [rngloc1].rhf;                                         03249
      tailf1 _ [rngloc1].rtf;                                        03250
      up1 _ getup(stid1);                                            03251
      pred1 _ getprd(stid1);                                         03252
      tailsub1 _ IF sub1 = stid1.stpsid THEN 0 ELSE getail(
      getsub(stid1));                                                03253
   % Save stid2 ring values and up and predecessor %                 03254
      lodent( stid2, rngtyp : rngloc2);                              03255
      IF [rngloc2].rsid = 0 THEN err($"Bad statement identifier");
                                                                     03256
      sub2 _ [rngloc2].rsub;                                         03257
      suc2 _ [rngloc2].rsuc;                                         03258
      hedf2 _ [rngloc2].rhf;                                         03259
      tailf2 _ [rngloc2].rtf;                                        03260
      up2 _ getup(stid2);                                            03261
      pred2 _ getprd(stid2);                                         03262
      tailsub2 _ IF sub2 = stid2.stpsid THEN 0 ELSE getail(
      getsub(stid2));                                                03263
   % Store stid1 ring values in stid2 ring (rngloc is still address
   of stid2 ring) %                                                  03264
      lodent( stid2, rngtyp : rngloc2 ); % Block may have gone out %
                                                                     03265
      IF [rngloc2].rsid = 0 THEN err($"Bad statement identifier");
                                                                     03266
      [rngloc2].rsub _  CASE sub1 OF                                 03267
         = stid1.stpsid: stid2.stpsid;                               03268
         = stid2.stpsid: stid1.stpsid;                               03269
         ENDCASE sub1;                                               03270
      [rngloc2].rsuc _ IF suc1 = stid2.stpsid THEN stid1.stpsid ELSE
      suc1;                                                          03271
      [rngloc2].rhf _ hedf1;                                         03272
      [rngloc2].rtf _ tailf1;                                        03273
      % Check head flag of stid1; fill in ring fields of up or
      predecessor %                                                  03274
         IF hedf1 THEN                                               03275
            BEGIN                                                    03276
            IF up1 # stid2 THEN stosub( up1, stid2); %stid1 was at
            head%                                                    03277
            END                                                      03278
         ELSE                                                        03279
            BEGIN                                                    03280
            IF pred1 # stid2 THEN stosuc( pred1, stid2 );            03281
            END;                                                     03282
      % Make successor field of tail of substructure of 1 (if any)
      point to 2 %                                                   03283
```

```
            IF tailsub1 # 0 AND tailsub1 # stid2 THEN                    03284
               stosuc( tailsub1, stid2 );                                03285
         % Store stid2 ring values in stid1 ring %                       03286
            lodent( stid1, rngtyp : rngloc1); % Block may have gone out %
                                                                         03287
            IF [rngloc1].rsid = 0 THEN err($"Bad statement identifier");
                                                                         03288
            [rngloc1].rsub _ CASE sub2 OF                                03289
               = stid1.stpsid: stid2.stpsid;                             03290
               = stid2.stpsid: stid1.stpsid;                             03291
               ENDCASE sub2;                                             03292
            [rngloc1].rsuc _ IF suc2 = stid1.stpsid THEN stid2.stpsid ELSE
            suc2;                                                        03293
            [rngloc1].rhf _ hedf2;                                       03294
            [rngloc1].rtf _ tailf2;                                      03295
            % Check head flag of stid2; fill in ring fields of up or
            predecessor %                                                03296
               IF hedf2 THEN                                             03297
                  BEGIN                                                  03298
                  IF up2 # stid1 THEN stosub( up2, stid1); %stid2 was at
                  head%                                                  03299
                  END                                                    03300
               ELSE                                                      03301
                  BEGIN                                                  03302
                  IF pred2 # stid1 THEN stosuc( pred2, stid1 );          03303
                  END;                                                   03304
            % Make successor field of tail of substructure of 2 (if any)
            point to 1 %                                                 03305
               IF tailsub2 # 0 AND tailsub2 # stid1 THEN                 03306
                  stosuc( tailsub2, stid1 );                             03307
         %Set up return values%                                          03308
            newsd1 _ stid2;  newsd2 _ stid1;                             03309
         END                                                             03310
      ELSE                                                               03311
         BEGIN %cross file transpose%                                    03312
         newsd1 _ ccopsta(stid1, levsuc, stid2, FALSE, 0);               03313
         upctbl(stid2, newsd1, stid2);                                   03314
         newsd2 _ ccopsta(stid2, levsuc, stid1, FALSE, 0);               03315
         upctbl(stid1, newsd2, stid1);                                   03316
         IF (tmpstd _ getsub(stid1)) # stid1 THEN                        03317
            cmovgro(newsd1, levdown, gethed(tmpstd), getail(tmpstd), FALSE,
            0);                                                          03318
         cdelgro(stid1, stid1, FALSE, 0);                                03319
         IF (tmpstd _ getsub(stid2)) # stid2 THEN                        03320
            cmovgro(newsd2, levdown, gethed(tmpstd), getail(tmpstd), FALSE,
            0);                                                          03321
         cdelgro(stid2, stid2, FALSE, 0);                                03322
         END;                                                            03323
      END                                                                03324
   ELSE                                                                  03325
      BEGIN                                                              03326
      IF getsub(stid1) # stid1 THEN err($"illegal transpose");           03327
      IF getsub(stid2) # stid2 THEN err($"illegal transpose");           03328
      newsd1 _ ctragro(stid1, stid1, stid2, stid2, filter, &vsptr:
      up1%garbage%, newsd2, up2%garbage%);                               03329
      END;                                                               03330
```

```
  %msfst();%                                                    03331
RETURN(newsd1, newsd2);                                         03332
END.  %%
```

```
                                                                        03333
(ctratex)   % GB: core NLS Transpose Text procedure %
PROCEDURE (bug1, bug2, bug3, bug4 % => no value %);                     02342
   % Procedure description                                              02343
      FUNCTION                                                          02344
         Transposes the text at 'bug1' through 'bug2' with the text at
         'bug3' through 'bug4'.                                         02345
      ARGUMENTS                                                         02346
         bug1 - TEXT POINTER - start of first text to transpose         02347
         bug2 - TEXT POINTER - end of first text to transpose           03623
         bug3 - TEXT POINTER - start of second text to transpose        03624
         bug4 - TEXT POINTER - end of second text to transpose          03625
      RESULTS                                                           02348
         none                                                           02349
      NON-STANDARD CONTROL                                              02350
         none                                                           02351
      GLOBALS                                                           02352
         none                                                           02353
      EXAMPLE                                                           02354
         ctratex($tp1, $tp2, $tp3, $tp4)                                02355
      %                                                                 02356
   % Declarations %                                                     02357
      REF bug1, bug2, bug3, bug4;                                       01216
     %msntx();%                                                         01217
   cldtxt(&bug1, &bug2);                                                01218
   cldtxt(&bug3, &bug4);                                                01219
   IF POS SF(bug1) = SF(bug3) THEN                                      01220
      IF POS bug1 < bug3 THEN                                           01221
         ST bug1 _ SF(bug1) bug1, bug3 bug4, bug2 bug3,                 01222
            bug1 bug2, bug4 SE(bug1)                                    01223
      ELSE                                                              01224
         ST bug1 _ SF(bug1) bug3, bug1 bug2, bug4 bug1,                 01225
            bug3 bug4, bug2 SE(bug1)                                    01226
   ELSE                                                                 01227
      BEGIN                                                             01228
      *lit* _ bug1 bug2;                                                01229
      ST bug1 _ SF(bug1) bug1, bug3 bug4, bug2 SE(bug1);                01230
      ST bug3 _ SF(bug3) bug3, *lit*, bug4 SE(bug3);                    01231
      END;                                                              01232
     %msftx();%                                                         01233
   RETURN;                                                              01234
   END.   %%
```

```
                                                                        02360
(trntst)   % LB: checks for legal transpose groups %
PROCEDURE (first1, first2, secnd1, secnd2 % => no value %);             02989
   % Procedure description                                              02990
      FUNCTION                                                          02991
         Checks that the groups specified by the transpose commands have
         legal bounds.  It does this by calling MOVTST to see if 'first1,2'
         are in the group bounded by 'secnd1,2' or 'secnd1,2' in the group
         bounded by 'first1,2'.                                         02992
      ARGUMENTS                                                         02993
         none                                                           02994
      RESULTS                                                           02995
         none                                                           02996
      NON-STANDARD CONTROL                                              02997
         none                                                           02998
      GLOBALS                                                           02999
         none                                                           03000
      EXAMPLE                                                           03001
         none                                                           03002
      %                                                                 03003
   % Declarations %                                                     03004
   movtst(first1, secnd1, secnd2);                                      01208
   movtst(first2, secnd1, secnd2);                                      01209
   movtst(secnd1, first1, first2);                                      01210
   movtst(secnd2, first1, first2);                                      01211
   RETURN;                                                              01212
   END.   %%
```

```
                                                                        03007
%trim%                                                                  01236
   (ctridir)   % UB: core NLS Trim connected Directory procedure %
   PROCEDURE (nver, astr % => no value %);                              02112
      % Procedure description                                          02113
         FUNCTION                                                       02114
            none                                                        02115
         ARGUMENTS                                                      02116
            nver - INTEGER - number of versions of each file to keep    02131
            astr - STRING - string to receive results of trim           02132
         RESULTS                                                        02118
            none                                                        02119
         NON-STANDARD CONTROL                                           02120
            none                                                        02121
         GLOBALS                                                        02122
            none                                                        02123
         EXAMPLE                                                        02124
            none                                                        02125
         %                                                              02126
      % Declarations %                                                 02127
         LOCAL                                                          01242
            fjfn,         % main jfn %                                  01243
            jfnflgs,      % left half flags for a group jfn %           01244
            pcjfn,        % jfn of partial copy %                       01245
            flags,        % bits indicating * typed for file name fields % 01246
            ntver;        % actual number of versions deleted %          01247
         LOCAL STRING                                                   01249
            jfnname[200],       % complete name from jfns %             01250
            errstring[200],     % eror message string %                 01251
            pstring[200];       % temporary string %                    01252
         REF astr;                                                      01254
      % initial variables %                                            01256
         fjfn _ jfnflgs _ pcjfn _ flags _ 0;                            01257
      % must keep 1 or more versions %                                 01258
         IF nver <= 0 THEN err( $"must keep 1 or more versions" );      01259
      % set up flags indicating file name of *.*;* %                   01260
         flags.fstar _ flags.estar _ flags.vstar _ TRUE;                01261
      % get main jfn (old file only, group jfn) %                      01262
         IF NOT (fjfn _ sgtjfn( gtjoif .V gtjstr + 0, $dirsfname, $errstring :
         jfnflgs ) )THEN err( $errstring );                             01263
      %set jfn flags to get highest version for TOPS20%                 03683
         IF tops20flag THEN                                             03684
            jfnflgs _ (jfnflgs .V 4000B) .A 767777B;                    03685
      % now loop to get all files in the group %                       01264
         LOOP                                                           01265
            BEGIN                                                       01266
            % check to make sure this not a PC with * for extension name %
                                                                        01267
               IF NOT chkpcs( fjfn, flags) THEN GOTO gnxtjfn;           01268
            % get actual file name into string for use later (maybe) %  01269
               r3 _ 011100B6 + 1;       % <directory>file.ext %        01270
               jfnflink( fjfn, $jfnname, r3);                           01271
            % if this is same as previous one then go to next one %     01272
               IF *jfnname* = *pstring* THEN GOTO gnxtjfn;               01273
               *pstring* _ *jfnname*;                                   01274
            % get PC jfn and find out if this user can access this PC %  01275
```

```
        IF NOT getpcjfn( fjfn, $pcjfn, $errstring, FALSE) THEN      01276
           BEGIN % this user can't access this file %               01277
           *astr* _ *astr*, "   ***   ", *jfnname*, " not trimmed
           because [", *errstring*, "], CR, LF;                     01278
           GOTO gnxtjfn;                                            01279
           END;                                                     01280
        % now delete the file (and its partial copy) %              01281
           IF NOT triflandpc( fjfn, pcjfn, nver, $errstring : ntver) THEN
                                                                    01282
           BEGIN % can't delete file (or partial copy) %            01283
           *astr* _ *astr*, "   ***   ", *jfnname*, " not trimmed
           because ", *errstring*, CR, LF;                          01284
           GOTO gnxtjfn;                                            01285
           END;                                                     01286
        % now tell the user if we have trimmed any files %          01287
           IF ntver > 0 THEN                                        01288
           BEGIN                                                    01289
           *astr* _ *astr*, "   ", *jfnname*, "  ", STRING( ntver ), "
           file";                                                   01290
           IF ntver > 1 THEN *astr* _ *astr*, "s;                   01291
           IF pcjfn THEN                                            01292
              IF ntver > 1 THEN *astr* _ *astr*, " (and their partial
              copies)"                                              01293
              ELSE *astr* _ *astr*, " (and it's partial copy)";     01294
           *astr* _ *astr*, " deleted.", CR, LF;                    01295
           END;                                                     01296
        % now get the next file in the group %                      01297
           (gnxtjfn):                                               01298
           % now get rid of the jfn for the partial %               01299
              IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                01300
           r1.LH _ jfnflgs;                                         01301
           r1.RH _ fjfn;                                            01302
           IF NOT SKIP !gnjfn( r1 ) THEN EXIT LOOP;                 01303
        END;                                                        01304
    % now get rid of any lingering jfns %                           01305
        IF NOT SKIP !rljfn( fjfn ) THEN NULL;                       01306
        IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                      01307
    % all done now, so return %                                     01308
      RETURN;                                                       01309
    END.   %%
```

```
                                                                        02130
%undelete%                                                              01313
   (cundarcfil)   % JB: core NLS Jndelete Archived File procedure %
   PROCEDURE (f1, f2 % => no value %);                                  03009
      % Procedure description                                          03010
         FUNCTION                                                       03011
            Not implemented yet.                                        03012
         ARGUMENTS                                                      03013
            none                                                        03014
         RESULTS                                                        03015
            none                                                        03016
         NON-STANDARD CONTROL                                           03017
            none                                                        03018
         GLOBALS                                                        03019
            none                                                        03020
         EXAMPLE                                                        03021
            none                                                        03022
         %                                                              03023
      % Declarations %                                                 03024
         REF f1, f2;                                                    01316
      err( notyet );                                                    01317
      RETURN;                                                           01318
      END.   %%
```

```
                                                                        03027
(cundfil)   % GB: undelete a group of files and partial copies %
PROCEDURE (rhstn, fname, astr % => no value %);                         02047
   % Procedure description                                              02048
      FUNCTION                                                          02049
         Undeletes the group of files specified by 'fname' on host
         'rhostn'.  The files should have previously been deleted, but no
         error will occur if they had not been.                         02050
      ARGUMENTS                                                         02051
         rhstn - INTEGER -                                              03642
            number of the host on which the files reside.  Only the local
            host ('lhostn') is currently implemented.                   03643
         fname - STRING -                                               03644
            file group name string in TENEX format.  Connected directory is
            assumed if a directory is not specified.  Examples:         03645
               "*.*"                                                    03646
               "<SMITH>*.PC"                                            03647
               "<SMITH>FOO.NLS;3"                                       03648
         astr - STRING - string to receive list of undeleted files      02068
      RESULTS                                                           02053
         none                                                           02054
      NON-STANDARD CONTROL                                              02055
         Error if                                                       02056
            host is not the local host                                  03658
            file doesn't exist                                          03657
      GLOBALS                                                           02057
         lhostn, gtjoif, gtjstr, gtjidi - read only                     02058
      EXAMPLE                                                           02059
         cundfil(lhostn, $"FOO.*", $string)                             02060
      %                                                                 02061
   % Declarations %                                                     02062
      LOCAL                                                             01326
         fjfn,          % main jfn %                                    01327
         jfnflgs,       % left half flags for a group jfn %             01328
         pcjfn,         % jfn of partial copy %                         01329
         flags;         % bits indicating * typed for file name fields % 01330
      LOCAL STRING                                                      01332
         uname[200],            % complete name as entered by user %    01333
         udirname[40],          % user input directory name %            01334
         ufilname[40],          % user input file name %                 01335
         uextname[40],          % user input extension name %            01336
         uverno[40],            % user input version number %            01337
         ulftovr[40],           % user input beyond version number %     01338
         jfnname[200],          % complete name from jfns %              01339
         errstring[200],        % eror message string %                  01340
         tstring[200];          % temporary string %                     01341
      REF fname, astr;                                                  01343
   % initial variables %                                                01345
      fjfn _ jfnflgs _ pcjfn _ flags _ 0;                               01346
   % find out if remote or local (disk) file %                          01347
      CASE rhstn OF                                                     01348
         = lhostn:  NULL;                                               01349
         ENDCASE err( $"remote file manipulations not implemented yet" );
                                                                        01350
   % now parse user input strings %                                    01351
      parseinput( &fname, $udirname, $ufilname, $uextname, $uverno,
```

```
    $ulttovr, $uname, $flags);                                          01352
  % get main jfn (old file only, ignore deleted, group jfn) %           01353
    IF NOT (fjfn _ sgtjfn( gtjoif .V gtjstr .V gtjidl + 0, $uname,
    $errstring : jfnflgs ) )THEN err( $errstring );                     01354
  % now find out if we support this type of file %                      01355
    IF NOT chkdev( fjfn ) THEN err( $"only disk files supported" );     01356
  % now loop to get all files in the group %                           01357
    LOOP                                                                01358
      BEGIN                                                             01359
      % if this file already undeleted, then go on to next one %        01360
        !gtfdb( fjfn, 1B6 + 1B, $r3);                                   01361
        IF NOT r3 .A 4B10 THEN GOTO gtnxtjfn;                           01362
      % check to make sure this not a PC with * for extension name %
                                                                        01363
        IF NOT chkpcs( fjfn, flags) THEN GOTO gtnxtjfn;                 01364
      % get actual file name into string for use later (maybe) %       01365
        r3 _ 011110B6     % directory file ext ver %                    01366
          + (IF jfnflgs.lhjb9  THEN 1B6 ELSE 0) % ;Protection %         01367
          + (IF jfnflgs.lhjb10 THEN 1B5 ELSE 0) % ;Account %            01368
          + (IF jfnflgs.lhjb11 THEN 4B4 ELSE 0) % ;T %                  01369
          + 1;            % with proper file punctuation %            01370
        jfnflink( fjfn, $jfnname, r3);                                  01371
      % get PC jfn and find out if this user can access this PC %      01372
        IF NOT getpcjfn( fjfn, $pcjfn, $errstring, FALSE) THEN          01373
          BEGIN % this user can't access this file %                    01374
          *tstring* _ *jfnname*, " not undeleted because ",
          *errstring*, ";                                               01375
          IF flags THEN                                                 01376
            BEGIN                                                       01377
            *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;           01378
            GOTO gtnxtjfn;                                              01379
            END                                                         01380
          ELSE                                                          01381
            BEGIN                                                       01382
            IF NOT SKIP !rljfn( fjfn) THEN NULL;                        01383
            err( $tstring );                                            01384
            END;                                                        01385
          END);                                                         01386
      % now undelete the file (and its partial copy) %                 01387
        IF NOT undflandpc( fjfn, pcjfn, $errstring) THEN                01388
          BEGIN % can't undelete file (or partial copy) %               01389
          *tstring* _ *jfnname*, " not undeleted because ",
          *errstring*;                                                  01390
          IF flags THEN                                                 01391
            BEGIN                                                       01392
            *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;           01393
            GOTO gtnxtjfn;                                              01394
            END                                                         01395
          ELSE                                                          01396
            BEGIN                                                       01397
            IF NOT SKIP !rljfn( fjfn) THEN NULL;                        01398
            IF NOT SKIP !rljfn( pcjfn) THEN NULL;                       01399
            err( $tstring );                                            01400
            END;                                                        01401
          END;                                                          01402
      % now tell the user we have undeleted this file %                 01403
```

```
            *astr* _ *astr*, "   ", *jfnname*;                              01404
            IF pcjfn THEN *astr* _ *astr*, " and its partial copy";          01405
            *astr* _ *astr*, CR, LF;                                         01406
          % now get rid of the jfn for the partial %                         01407
            IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                           01408
          % now get the next file in the group %                             01409
            (gtnxtjfn):                                                      01410
            r1.LH _ jfnflgs;                                                 01411
            r1.RH _ fjfn;                                                    01412
            IF NOT SKIP !gnjfn( r1 ) THEN EXIT LOOP;                         01413
          END;                                                               01414
      % now get rid of any lingering jfns %                                  01415
        IF NOT SKIP !rljfn( fjfn ) THEN NULL;                                01416
        IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                               01417
      % all done now, so return %                                           01418
       RETURN;                                                               01419
      END.   %%
```

```
                                                                    02065
(cundmodfil)   % JB: core NLS Jndelete Modifications to File proc %
PROCEDURE (fileno % => no value %);                                 02028
   % Procedure description                                          02029
      FUNCTION                                                      02030
         Undeletes the Partial Copy associated with the NLS file 'fileno',
         thereby restoring the modifications contained in that partial
         copy.  Then updates the file header to replect the fact that the
         file is again locked (has a PC).                           02031
      ARGUMENTS                                                     02032
         fileno - INTEGER - NLS file number of the file             02033
      RESULTS                                                       02034
         none                                                       02035
      NON-STANDARD CONTROL                                          02036
         Error if the partial copy is not deleted or if undeleting it
         fails.                                                     02037
      GLOBALS                                                       02038
         none                                                       02039
      EXAMPLE                                                       02040
         cundmodfil(stid.stfile)                                    02041
      %                                                             02042
   % Declarations %                                                 02043
      LOCAL fl;  REF fl;                                            03678
      LOCAL STRING errstr[100];                                     03674
      &fl _ flntadt(fileno);                                        03679
      IF fl.flpart THEN err($"Partial copy is already undeleted.");  03680
   writeout(fileno);                                                03675
   IF NOT makepc(&fl, fileno, TRUE, $errstr) THEN                   03676
      err($errstr);                                                 03677
   RETURN;                                                          01426
   END.   %%
```

```
                                                                        02046
%update file%                                                           01428
   (cupdfil)   % GB: core NLS Update File procedure %
   PROCEDURE (fileno, type, namstr % => no value %);                    02009
      % Procedure description                                           02010
         FUNCTION                                                       02011
            Updates the file with file number 'fileno' according to the type
            of update specified.  The string 'namstr' should contain the new
            file name if Update File Rename is done.                    02012
         ARGUMENTS                                                      02013
            fileno - INTEGER - NLS file number of the file              02014
            type - INTEGER - type of update                             03659
               newversion => update to next higher version              03661
               oldversion => update to old (current) version            03660
               newname    => update to next higher version of name in 'namstr'
                                                                        03662
               upcompact  => compacts the file and updates to next higher
               vesion                                                   03663
            namstr - STRING - new file name, or 0; no *'s allowed       03664
         RESULTS                                                        02015
            none                                                        02016
         NON-STANDARD CONTROL                                           02017
            Error if illegal type of update or if update is unnecesary (e.g.
            file is not locked)                                         02018
         GLOBALS                                                        02019
            newversion, oldversion, newname, upcompact - read only      02020
         EXAMPLE                                                        02021
            cupdfil(stid.stfile, newversion, 0)                         02022
            cupdfil(stid.stfile, newname, $"FOO.NLS")                   03665
         %                                                              02023
      % Declarations %                                                  02024
      CASE type OF                                                      01431
         = oldversion, = newversion:                                    01432
            IF NOT updtfl (fileno, type, 0) THEN                        01433
               err($"Update unnecessary: no modifications");            01434
         = newname:                                                     01435
            IF NOT updtfl (fileno, newname, namstr) THEN                01436
               err($"Update unnecessary: no modifications");            01437
         = upcompact:                                                   01438
            compactfile (fileno);                                       01439
         ENDCASE err($"Illegal update request");                        01440
      RETURN;                                                           01441
      END.   %%
```

```
                                                                        02027
(updtfl)   % LB: does Update File %
PROCEDURE (fileno, vertype, filname % => TRUE or FALSE %);              03028
   % Procedure description                                              03029
      FUNCTION                                                          03030
         none                                                           03031
      ARGUMENTS                                                         03032
         fileno - INTEGER - NLS file number of file to be updated       03047
         vertype - INTEGER -                                            03048
            newversion if Update new, oldversion if update old          03053
         filname - STRING -                                             03049
            new name of updated file.  May be zero if we want to use the
            current name which is available through the file status table.
                                                                        03054
      RESULTS                                                           03034
         procedure-value - BOOLEAN -                                    03035
            TRUE if file updated; FALSE if no changes, file not updated.
                                                                        03052
      NON-STANDARD CONTROL                                              03036
         none                                                           03037
      GLOBALS                                                           03038
         none                                                           03039
      EXAMPLE                                                           03040
         none                                                           03041
      %                                                                 03042
   % Declarations %                                                     03043
      LOCAL                                                             01450
         fl, dt, fhd, stid, flags, rn, stdb, sdbblk, rngblk, hd, hdblk,
         hedb, pgindx, lokdir, lokinit, newjfn, fhdpag, count;          01451
      LOCAL STRING nfname[200], locstr[200], datestr[40];               01452
      LOCAL TEXT POINTER ptr;                                           01453
      REF fl, dt, rn, hd, filname;                                      01454
   % get address of file status table entry for this file %             01455
      &fl _ flntadr(fileno);                                            01456
   % get file link for the old file %                                   01457
      jfnflink( fl.florig, $locstr, 011110B6+1);                        01458
    %used to call ckdiracc here for ARC TENEX 131%                      01459
   % if file in browse mode, put out error message, don't update. %    01462
      IF fl.flbrws THEN                                                 01463
         BEGIN                                                          01464
         *lit* _ *[fl.flastr]*, " is in browse mode";                   01465
         err($lit);                                                     01466
         END;                                                           01467
   IF fl.flexis AND fl.florig THEN                                      01468
      IF fl.flpart OR (vertype = newversion AND &filname)  OR (vertype =
      newname AND &filname)THEN                                         01469
         BEGIN                                                          01470
         IF vertype = newversion OR vertype = newname THEN %open new
         version with write access%                                     01471
            BEGIN                                                       01472
            IF &filname THEN                                            01473
               BEGIN                                                    01474
               IF NOT (newjfn _ lgetjfn                                 01475
                  (0,&filname, $nlsext, gtjoof, $lit)) THEN             01476
                     err($lit);                                         01477
               jfnstr(newjfn, $nfname);                                 01478
```

```
            IF NOT sysopen(newjfn, readwrite, random, $lit) THEN    01479
               BEGIN                                                01480
               reljfn(newjfn);                                      01481
               err($lit);                                           01482
               END;                                                 01483
            END)                                                    01484
         ELSE                                                       01485
            BEGIN                                                   01486
            *nfname* _ *[fl.flastr]*;                               01487
            count _ nfname.L;                                       01488
            WHILE *nfname*[count] NOT= fvrdchar DO                  01489
               count _ count-1; %fvrdchar is ";" for tenex, "." for
               tops20%                                              01490
            nfname.L _ count-1; %delete version number from name%   01491
            flags _ getgtjflg(write, origff, dfltvrs);              01492
            IF NOT (newjfn _ sgtjfn(flags, $nfname, $lit)) THEN     01493
               err($lit)                                            01494
            ELSE IF NOT sysopen(newjfn, readwrite, random, $lit) THEN
                                                                    01495
               BEGIN                                                01496
               reljfn(newjfn);                                      01497
               err($lit);                                           01498
               END;                                                 01499
            EN)                                                     01500
         END                                                        01501
      ELSE %close and reopen old version with write access%         01502
         BEGIN                                                      01503
         IF NOT writeout(fileno, $lit)
         OR NOT sysclose(fl.florig .V 4B11, $lit) THEN              01504
            err($lit);                                              01505
         IF NOT sysopen(fl.florig, readwrite, random, $lit)         01506
         THEN                                                       01507
            BEGIN                                                   01508
            %failed to get write access, reopen with read access%   01509
            dismes(2, $lit);                                        01510
            IF NOT rdhdr(fileno, $lit) THEN err($lit);              01511
            IF NOT sysopen(fl.florig, read, random, $lit) THEN      01512
               err($lit);                                           01513
            GOTO STATE;                                             01514
            END;                                                    01515
         newjfn _ fl.florig;                                        01516
         IF NOT rdhdr(fileno, $lit) THEN err($lit);                 01517
         IF lit.L > empty THEN                                      01518
            BEGIN                                                   01519
            dismes(2, $lit);                                        01520
            END;                                                    01521
         END;                                                       01522
      % get partial copy header address %                          01523
         fhd _ filehead[fileno];                                    01524
      % first do the ring blocks %                                 01525
         &rn _ fhd - $filhed + $rngst;                              01526
         stid _ 0;                                                  01527
         stid.stfile _ fileno;                                      01528
         rngblk _ 0;                                                01529
         DO                                                         01530
            BEGIN                                                   01531
```

```
            IF rn.rfexis THEN                                                   01532
               BEGIN                                                            01533
               IF (rn.rfpart OR vertype = newversion OR vertype =
               newname) THEN                                                    01534
                  BEGIN                                                         01535
                  stid.stblk _ rngblk;                                          01536
                  pgindx _ lodent(stid, rngtyp);                                01537
                  % copy page to the file %                                     01538
                  updtpg(rngblk + rngbas, pgindx, newjfn);                      01539
                  rn.rfcore _ 0;                                                01540
                  END;                                                          01541
               END;                                                             01542
            &rn _ &rn + 1;                                                      01543
            END                                                                 01544
         UNTIL (rngblk _ rngblk + 1) = rngm;                                    01545
      % now do the sdb blocks %                                                 01546
         stdb _ 0;                                                              01547
         stdb.stfile _ fileno;                                                  01548
         &dt _ fhd - $filhed + $dtbst;                                          01549
         sdbblk _ 0;                                                            01550
         DO                                                                     01551
            BEGIN                                                               01552
            IF dt.rfexis THEN                                                   01553
               BEGIN                                                            01554
               IF (dt.rfpart OR vertype = newversion OR vertype =
               newname) THEN                                                    01555
                  BEGIN                                                         01556
                  stdb.stblk _ sdbblk;                                          01557
                  pgindx _ lodent(stdb, sdbtyp);                                01558
                  % map in the file page %                                      01559
                  updtpg(sdbblk + dtbbas, pgindx, newjfn);                      01560
                  dt.rfcore _ 0;                                                01561
                  END;                                                          01562
               END;                                                             01563
            &dt _ &dt + 1;                                                      01564
            END                                                                 01565
         UNTIL (sdbblk _ sdbblk+1) = dtbm;                                      01566
      % extra header pages %                                                    03698
         hdblk _ 1; %for now, this is the only one%                             03699
            BEGIN                                                               03700
            hedb _ 0;                                                           03701
            hedb.stfile _ fileno;                                               03702
            &hd _ fhd - $filhed + $rfbs + hdblk;                                03703
            IF hd.rfexis AND (hd.rfpart OR vertype = newversion OR
            vertype = newname) THEN                                             03704
               BEGIN                                                            03705
               hedb.stblk _ hdblk;                                              03706
               pgindx _ lodent(hedb, hdtyp);                                    03707
               updtpg(hdblk+hedbas, pgindx, newjfn);                            03708
               hd.rfcore _ 0;                                                   03709
               END;                                                             03710
            END;                                                                03711
      % finally do the header %                                                 01567
         % map in the new file header %                                         01568
            r1.LH _ newjfn;                                                     01569
            r1.RH _ 0;                                                          01570
```

```
        r2.LH _ 4B5;                                                    01571
        r2.RH _ $ckpag / 1000B;                                         01572
        r3 _ 14B10;                                                     01573
        !JSYS pmap;                                                     01574
      % get addr of file header (including type info) %                 01575
        pgindx _ fl.flhead;                                             01576
        fhdpag _ crpgad[pgindx];                                        01577
      % block transfer old file header to new %                         01578
        movefof (fhdpag, $ckpag, 512);                                  01579
      %remove the old header%                                           01580
        r1 _ -1;                                                        01581
        r2.LH _ 4B5;                                                    01582
        r2.RH _ fhdpag / 512;                                           01583
        r3 _ 0;                                                         01584
        !JSYS pmap;                                                     01585
      fhd _ $ckpag + fbhdl;                                             01586
      % update last write time %                                        01587
        [fhd - $filhed + $lwtim] _ gtadcall();                          01589
      % update last write identification %                              01590
        [fhd - $filhed + $finit] _ cinit;                               01591
      %set partial copy bits in new header for subsequent updating of
      origin statemet%                                                  01592
        &rn _ fhd - $filhed + $rngst;                                   01593
        rngblk _ 0;                                                     01594
        DO                                                              01595
          BEGIN                                                         01596
          IF rn.rfexis THEN                                             01597
            BEGIN                                                       01598
            %pretend like pc for later edit of origin statement%
                                                                        01599
              rn.rfpart _ TRUE;                                         01600
            END;                                                        01601
          &rn _ &rn + 1;                                                01602
          END                                                           01603
        UNTIL (rngblk _ rngblk + 1) = rngm;                             01604
        &dt _ fhd - $filhed + $dtbst;                                   01605
        sdbblk _ 0;                                                     01606
        DO                                                              01607
          BEGIN                                                         01608
          IF dt.rfexis THEN                                             01609
            BEGIN                                                       01610
            %pretend like pc for later edit of origin statement%
                                                                        01611
              dt.rfpart _ TRUE;                                         01612
            END;                                                        01613
          &dt _ &dt + 1;                                                01614
          END                                                           01615
        UNTIL (sdbblk _ sdbblk+1) = dtbm;                               01616
       % remove the page %                                              01617
        r1 _ -1;                                                        01618
        r2.LH _ 4B5;                                                    01619
        r2.RH _ $ckpag / 1000B;                                         01620
        r3 _ 0;                                                         03670
        !JSYS pmap;                                                     01621
      % open the original file with read access %                       01622
        IF (vertype = newversion OR vertype = newname) THEN             01623
```

```
      BEGIN                                                          01624
      %Propogate  access%                                            01625
      IF vertype = newversion THEN                                   01626
         BEGIN                                                       01627
      %read old access%                                              01628
         r1 _ fl.florig;                                             01629
         r2 _ 1000024B;                                              01630
         r3 _ $r3;                                                   01631
         !JSYS gtfdb;                                                01632
      %propogate to new file%                                        01633
         r1 _ 24B6 .V newjfn;                                        01634
         r2 _ 0;                                                     01635
         r2.acctyp _ 36M;                                            01636
         !JSYS chfdb;                                                01637
         END;                                                        01638
      %propagate archive "don't archive" and "don't delete" bits%
                                                                     03200
         IF vertype # newname THEN                                   03201
            BEGIN                                                    03202
            !gtfdb(fl.florig, 1000017B, $r3);                        03204
            !chfdb(17B6 .V newjfn, 11B10); %r3 set from gtfdb%
                                                                     03205
            END;                                                     03203
      IF NOT <IOEXEC, writeout>(fileno, $lit)                        01639
      OR NOT <IOEXEC, delpc>(fileno, $lit)                           01640
      OR NOT <IOEXEC, sysclose>(fl.florig, $lit) THEN                01641
         err($lit);                                                  01642
      fl.flpart _ newjfn; %to fake out write psi stuff, in
      updating origin, below%                                        01643
      filepart[fileno] _ TRUE;                                       01644
      END                                                            01645
   ELSE                                                              01646
      BEGIN                                                          01647
      IF NOT <IOEXEC, writeout>(fileno, $lit)                        01648
      OR NOT <IOEXEC, delpc>(fileno, $lit) THEN                      01649
         err($lit);                                                  01650
      fl.flpart _ fl.florig; %to fake out write psi stuff, in
      updating origin, below%                                        01651
      filepart[fileno] _ TRUE;                                       01652
      END;                                                           01653
   IF NOT rdhdr(fileno, $lit) THEN err($lit);                        01654
   % update the origin statement and filst %                        01655
      fl.florig _ 0;                                                 01656
      *nfname* _ NULL;                                               01657
      jfnflink(fl.flpart,$nfname,011110B6+1);                        01658
      uplsfnm($locstr, $nfname);                                     01659
         %update file names in link stack%                           01660
      *datestr* _ NULL;                                              01668
      <AJXCOD, getdat>($datestr);                                    01669
      ptr _ orgstid;                                                 01670
      ptr.stfile _ fileno;                                           01671
      ptr[1] _ 1;                                                    01672
      setrot($ptr, $nfname, $datestr);                               01673
      *nfname* _ NULL;                                               01674
      jfnstr( fl.flpart, $nfname);                                   01675
      IF fl.flastr THEN freestring( fl.flastr := 0, $dspblk);
```

```
                                                                    01676
            fl.flastr _ getstring( nfname.L + 1, $dspblk);          01677
            *[fl.flastr]* _ *nfname*;                               01678
            IF fl.flpcst THEN freestring( fl.flpcst := 0, $dspblk);
                                                                    01680
            fl.flpcst _ cpcnam(fl.flastr, logdirno);                01681
            fldirectory(&fl);                                       01682
         % finish cleaning up %                                     01683
            dumpc(fileno, FALSE); %turn off pc bits in file%        01684
            IF NOT <IOEXEC, writeout>(fileno, $lit)                 01685
            OR NOT <IOEXEC, sysclose>(fl.flpart .V 4B11, $lit) THEN
                                                                    01686
               err($lit);                                           01687
            fl.florig _ fl.flpart := 0;                             01688
            filepart[fileno] _ FALSE;                               01689
            IF NOT sysopen(fl.florig, read, random, $lit) THEN      01690
               err($lit);                                           01691
         %delete old versions%                                      01692
            delovsrns(fl.florig, nvtk);                             01693
         IF NOT rdhdr(fileno, $lit) THEN err($lit);                 01694
         IF lit.L > empty THEN                                      01695
            BEGIN                                                   01696
            dismes(2, $lit);                                        01697
            END;                                                    01698
      END                                                           01699
   ELSE %check to see if it is locked without a partial copy%       01700
      %

      There is some danger in this.  If the file is being accessed by
      the same user at another console we are going to unlock the file
      out from under him.

      %                                                             01701
      BEGIN                                                         01702
      r1 _ fl.florig;                                               01703
      r2 _ 10000248;                                                01704
      r3 _ $r3;                                                     01705
      !JSYS gtfdb;                                                  01706
      lokdir _ r3.lkdirn;                                           01707
      lokinit _ r3.lkinit;                                          01708
      !JSYS gjinf;                                                  01709
      IF r1 = lokdir AND lokinit = cinit THEN                       01710
         BEGIN  %file is locked by this user...unlock it%           01711
         IF NOT maywrt(fileno) THEN err($"No write access");        01712
         r1.LH _ 248;                                               01713
         r1.RH _ fl.florig;                                         01714
         r2 _ 0;                                                    01715
         r2.lkinit _ r2.lkdirn _ 36M;                               01716
         r3 _ 0;                                                    01717
         !JSYS chfdb;                                               01718
         END                                                        01719
      ELSE                                                          01720
         RETURN(FALSE); %Update unnecessary%                        01721
      END;                                                          01722
RETURN(TRUE); %File has been updated%                               01723
END.  %%
```

```
                                                                        03046
(updtpg)   % LB: maps pages for Update File %
PROCEDURE (fpage, pgindx, newjfn % => no value %);                      03055
   % Procedure description                                              03056
      FUNCTION                                                          03057
         Used in update to map a page from the destination file into the
         update workarea page (beginning at ckpag).  Then block transfers a
         page from the original file or the PC to the workarea.  Finally
         puts this page back into the destination file page by pmapping it
         back out (TENEX page tables keep track of this).  Page is removed
         from NLS core page status table.                               03058
      ARGUMENTS                                                         03059
         fpage - INTEGER -                                              03074
            destination page in the file being constructed which we are
            working on                                                  03079
         pgindx - INTEGER -                                             03076
            index into core page status table of page in file (original or
            partial copy) to be sent to the destination file            03080
         newjfn - INTEGER - jfn of the file being updated to            03078
      RESULTS                                                           03061
         none                                                           03062
      NON-STANDARD CONTROL                                              03063
         none                                                           03064
      GLOBALS                                                           03065
         none                                                           03066
      EXAMPLE                                                           03067
         none                                                           03068
      %                                                                 03069
   % Declarations %                                                     03070
      LOCAL blk, ct;                                                    01732
      REF ct;                                                           01733
   % map in the destination file page%                                  01734
      r1.LH _ newjfn;                                                   01735
      r1.RH _ fpage;                                                    01736
      r2.LH _ 4B5;                                                      01737
      r2.RH _ $ckpag / 1000B;                                           01738
      r3 _ 14B10;                                                       01739
      !JSYS pmap;                                                       01740
   % get the address of the old file page (which is in core) to be sent to
   the destination %                                                    01741
      &ct _ $corpst + pgindx;                                           01742
      blk _ crpgad[pgindx];                                             01743
   % block transfer the from the old location to the workarea %         01744
      mvbtbf (blk, $ckpag, 512);                                        01745
   %remove sent page from core-- doing this sends it to the destination
   file%                                                                01746
      r1 _ -1;                                                          01747
      r2.LH _ 4B5;                                                      01748
      r2.RH _ blk / 512;                                                01749
      r3 _ 0;                                                           01750
      !JSYS pmap;                                                       01751
   % Remove all traces of it from the NLS core page status table %      01752
      ct _ 0;                                                           01753
   RETURN;                                                              01754
   END.   %%
```

```
                                                                        03073
(setrot)   % LB: used to build statements %
PROCEDURE (ptr, fnstng, dtstng % => no value %);                        03081
   % Procedure description                                              03082
      FUNCTION                                                          03083
         none                                                           03084
      ARGUMENTS                                                         03085
         none                                                           03086
      RESULTS                                                           03087
         none                                                           03088
      NON-STANDARD CONTROL                                              03089
         none                                                           03090
      GLOBALS                                                           03091
         none                                                           03092
      EXAMPLE                                                           03093
         none                                                           03094
      %                                                                 03095
   % Declarations %                                                     03096
      REF ptr, fnstng, dtstng;                                          01758
   FIND SF(ptr) ^p1 ^p2 ["";;;;"] ^p2;                                  01759
   IF FIND < ["" >>>"] > 4CH ^p1                                        01760
   THEN ST p1 p2 _ *fnstng*, ", " , *dtstng*, SP, *initsr*, " ;;;;"     01761
   ELSE                                                                 01762
      IF NOT gethmf(ptr)                                                01763
      THEN ST ptr _ *fnstng*, ", " , *dtstng*, SP, *initsr*, " ;;;;", p2
      SE(ptr);                                                          01764
   RETURN;                                                              01765
   END.   %%
```

```
                                                                        03099
(compacttile)   % LB: does Update File Compact %
PROCEDURE (ofn % => no value %);                                        03100
   % Procedure description                                              03101
      FUNCTION                                                          03102
         none                                                           03103
      ARGUMENTS                                                         03104
         ofn - INTEGER -                                                03105
            old NLS file number (of the file being output)              03120
      RESULTS                                                           03106
         none                                                           03107
      NON-STANDARD CONTROL                                              03108
         none                                                           03109
      GLOBALS                                                           03110
         none                                                           03111
      EXAMPLE                                                           03112
         none                                                           03113
      %                                                                 03114
   % Declarations %                                                     03115
      LOCAL njfn, nfn, nfl, ofl, temp, oprvsts, curskpachk;             01770
      LOCAL TEXT POINTER tp;                                            01771
      LOCAL STRING nfilnam[200], locstr[200];                           01772
      REF nfl, ofl;                                                     01773
   curskpachk _ skpachk;                                                01774
   ON SIGNAL ELSE skpachk _ curskpachk;                                 01775
   skpachk _ TRUE;                                                      01776
   &ofl _ flntadr(ofn);                                                 01777
   oprvsts _ rdprvsts (ofn);                                            01778
   %used to call ckdiracc here for ARC TENEX 131%                       03199
   *nfilnam* _ *[ofl.flastr]*;                                          01781
   incrvrsn($nfilnam);                                                  01782
   IF NOT (njfn _ lgetjfn (0, $nfilnam, $nlsext, gtjoof, $lit)) THEN    01783
      SIGNAL (ofilerr, $lit);                                           01784
   jfnstr (njfn, $nfilnam);   % get full file name %                    01785
   nfn _ addfil (njfn, $nfilnam);                                       01786
   %make it a partial copy%                                             01787
      &nfl _ flntadr (nfn);                                             01788
      nfl.flpart _ nfl.florig;                                          01789
      filepart[nfn] _ TRUE;                                             01790
   IF NOT sysopen (nfl.flpart, readwrite, random, $lit) THEN            01791
      SIGNAL (ofilerr, $lit);                                           01792
   clsid($clhead); %convert psid's to sid's in correspondence list%     01793
   copfil (ofn, nfn, $clhead);                                          01794
   %update the new origin statement%                                    01795
      *lit* _ NULL;                                                     01796
      <AUXCOD, dtfrmt> ([filhdr (nfn) + $lwtim - $filhed], $lit);       01797
      tp _ 0;                                                           01798
      tp.stfile _ nfn;                                                  01799
      tp.stpsid _ origin;                                               01800
      tp[1] _ 1;                                                        01801
      % get file name in link form %                                    01802
         jfnflink( nfl.florig, $locstr, 011110B6+1);                    01803
      setrot ($tp, $locstr, $lit);                                      01804
      IF nfl.flastr THEN freestring( nfl.flastr := 0, $dspblk);         01805
      nfl.flastr _ getstring( nfilnam.L + 1, $dspblk);                  01806
      *[nfl.flastr]* _ *nfilnam*;                                       01807
```

```
   IF nfl.flpcst THEN freestring( nfl.flpcst := 0, $dspblk);          01810
   nfl.flpcst _ cpcnam (nfl.flastr, logdirno);                        01811
%propagate privacy status%                                            01812
   chprvsts (nfn, oprvsts);                                           01813
% propagate archive "don't archive" and "don't delete%                01206
   !gtfdb(ofl.florig, 1000017B, $r3);                                 03207
   !chfdb(17B6 .V njfn, 11B10); %r3 set by gtfdb%                     03208
IF NOT writeout(ofn,$lit) OR NOT delpc (ofn, $lit) THEN err ($lit);
                                                                      01814
IF lit.L > empty THEN                                                 01815
   BEGIN                                                              01816
   dismes (2, $lit);                                                  01817
   END;                                                               01818
   %delete partial copy from user's directory and unlock original file
   (if it exists)%                                                    01819
% close old file %                                                    01820
   dumpc(nfn, FALSE);  %turn off pc bits in header%                   01821
   IF NOT clsfil (ofn, $lit) THEN err ($lit);                         01822
%make new file the original%                                          01823
   nfl.florig _ nfl.flpart := 0;                                      01824
   filepart[nfn] _ FALSE;                                             01825
% make new file use old file number %                                 01826
   % write out all pages from the file %                              01827
   IF NOT writeout (nfn, $lit) THEN err ($lit);                       01828
   cpyfle (nfn, ofn);                                                 01829
      %copies new file list entry to old file list entry%             01830
   clrfil (nfn); %clear  file list entry for new file%                01831
   IF NOT rdhdr(ofn, $lit) THEN err ($lit);                           01972
   clpsid($clhead); %convert new sid's back to psid's in correspondence
   list%                                                              01973
   clupdt();  %done here so file header can be mapped out after markers
   are updated per new psid's%                                        01974
   r1.LH _ 4B5;  r1.RH _ filhdr(ofn)/512;                             01975
   r2.LH _ njfn;  r2.RH _ 0;                                          01976
   r3 _ 0;                                                            03671
   !JSYS pmap;                                                        01977
   r2 _ r1;                                                           01978
   r1 _ -1;                                                           01979
   !JSYS pmap;                                                        01980
   % close and re-open outputed file with read-only                   01832
   status %                                                           01833
      IF NOT clsfil (ofn, $lit) THEN err ($lit);                      01834
      IF NOT opnfil (ofn, $lit) THEN err ($lit);                      01835
      IF lit.L > empty THEN                                           01836
         BEGIN                                                        01837
         dismes (2, $lit);                                            01838
         END;                                                         01839
%delete old versions%                                                 01841
   delovsrns ([flntadr(ofn)].florig, nvtk);                           01842
skpachk _ curskpachk;                                                 01843
RETURN;                                                               01844
END.   %%
```

```
                                                                        03118
% update and compact file support routines %                            01846
   (coptil)   % LR: copies and compacts file blocks %
   PROCEDURE (oldfil, newfil, list % => no value %);                    03121
      % Procedure description                                           03122
         FUNCTION                                                       03123
            Copies and compacts the ring and data blocks of the file
            identified by oldfil to the file identified by newfil, one
            statement at a time, using cisid to convert the entries in the
            correspondence list to sid's and clpsid to convert the sid's to
            the new stid's.  The correspondence list is asssumed to contain
            stid's of interest, such as the csp's and frozen list stid's
            for each da, entries on the link-jump rings, and markers for
            this file.                                                  03144
            (used by outfile)                                           03145
         ARGUMENTS                                                      03125
            oldfil - INTEGER -                                          03141
               NLS file number of the old file being output             03146
            newfil - INTEGER -                                          03142
               NLS file number of the new file being output to          03147
            list - ? - address of correspondence list head              03143
         RESULTS                                                        03127
            procedure-value                                             03128
         NON-STANDARD CONTROL                                           03129
            none                                                        03130
         GLOBALS                                                        03131
            none                                                        03132
         EXAMPLE                                                        03133
            none                                                        03134
         %                                                              03135
      % Declarations %                                                  03136
         LOCAL                                                          01854
            lstid,     %last statement identifier%                      01855
            ostid,     %old statement identifier%                       01856
            nstid,     %new statement identifier%                       01857
            nhd,       %new file header address%                        01858
            ohd,       %old file header address%                        01859
            mkrlen,    %number of entries in marker table%              01860
            oosdb,     %old origin sdb%                                 01861
            dir;       %direction%                                      01862
         LOCAL TEXT POINTER z1, z2, z3, z4;                             01863
         LOCAL STRING nfnam[75], ofnam[75], dftdirnam[75];              01864
         REF list, oosdb;                                               01865
      ostid _ nstid _ lstid _ 0;                                        01866
      ostid.stfile _ oldfil;                                            01867
      lstid.stfile _ nstid.stfile _ newfil;                             01868
      ostid.stpsid _ lstid.stpsid _ nstid.stpsid _ origin;              01869
      inttil(newfil);                                                   01870
         %initialize new file header%                                   01871
      LOOP  %copy statements from old file to new%                      01872
         BEGIN                                                          01873
         IF (nstid _ getsub(ostid)) NOT= ostid THEN                     01874
            BEGIN  %statement has sub structure%                        01875
            ostid _ nstid;                                              01876
            dir _ sub;                                                  01877
            END                                                         01878
```

```
      ELSE  %no substructure--check if tail%                              01879
         BEGIN                                                            01880
         IF ostid.stpsid = origin THEN EXIT;                              01881
         IF NOT getftl(ostid) THEN                                        01882
            BEGIN  %not tail--get successor%                              01883
            ostid _ getsuc(ostid);                                        01884
            dir _ suc;                                                    01885
            END                                                           01886
         ELSE  %tail--get successor of highest 'up' proccessed%           01887
            BEGIN                                                         01888
            DO                                                            01889
               BEGIN  %get 'up' until not tail%                           01890
               IF (ostid _ getup(ostid)).stpsid = origin                  01891
               THEN                                                       01892
                  EXIT 2;                                                 01893
               lstid _ getup(lstid);                                      01894
               END                                                        01895
            UNTIL NOT getftl(ostid);                                      01896
            dir _ suc;                                                    01897
            ostid _ getsuc(ostid);                                        01898
            END;                                                          01899
         END;                                                             01900
      nstid _ newrng(newfil);                                             01901
         %allocate new ring block in new file%                            01902
      coprng(ostid, nstid);                                               01903
         %copy pertinent feilds from old ring block to new%               01904
      %insert new ring element into new file                              01905
         as suc or sub of lstid%                                          01906
      IF dir = suc THEN <STRMNP, inss> (lstid, nstid, nstid)              01907
      ELSE <STRMNP, insd> (lstid, nstid, nstid);                          01908
      copplist(ostid, nstid);                                             01909
         %copy contents of oldsdb to new%                                01910
      lstid _ nstid;                                                      01911
      END;                                                                01912
   *ofnam* _ NULL;                                                        01913
   *nfnam* _ NULL;                                                        01914
   filnam(oldfil, $ofnam);                                                01915
   filnam(newfil, $nfnam);                                                01916
   uplsfnm($ofnam, $nfnam); %update file name in link stack%              01917
   %replace origin statement%                                             01918
      p1 _ p2 _ 0;                                                        01919
      p1.stfile _ oldfil;                                                 01920
      p2.stfile _ newfil;                                                 01921
      p1.stpsid _ p2.stpsid _ origin;                                     01922
      p1[1] _ p2[1] _ 1;                                                  01923
      FIND SF(p2) ^z1 SE(p2) ^z2 SF(p1) ^z3 SE(p1) ^z4;                   01924
      creptex($z1, $z2, $z3, $z4);                                        01925
      stosid(p2, getsid(p1));                                             01926
      lodprop(p1, txttyp : &oosdb); %get old origin sdb and%              01927
      csetnsta(p2, oosdb.slnmdl, oosdb.srnmdl); %copy over name delims%
                                                                          01928
   %copy sidcnt from old header%                                          01929
      [(nhd _ filhdr(newfil)) + $sidcnt-$filhed] _                        01930
         [(ohd _ filhdr(oldfil)) + $sidcnt-$filhed];                      01931
   %copy default link directory from old header%                          01932
      gdftdir(oldfil, $dftdirnam);                                        03686
```

```
   FIND SF(*dftdirnam*) ^z1 SE(z1) ^z2;                                03687
   csetlindef(newfil, $z1, $z2);                                       03688
%copy marker table%                                                    01935
   mkrlen _ [ohd + $mkrtbl-$filhed];                                   01936
   [nhd + $mkrtbl-$filhed] _ mkrlen;                                   01937
   IF mkrlen > 0 THEN                                                  01938
      mvofof(ohd + $mkrtb-$filhed, nhd + $mkrtb-$filhed,
      mkrlen*mkrl);                                                    01939
%set lwtim, finit%                                                     01940
   [nhd + $finit - $filhed] _ cinit;                                   01941
   [nhd + $lwtim - $filhed] _ gtadcall();                              01943
RETURN;                                                                01944
END.   %%
```

```
                                                                        03139
(coprng)   % LB: copies ring-element fields %
PROCEDURE (ostid, nstid % => no value %);                               03149
   % Procedure description                                              03150
      FUNCTION                                                          03151
         Copies name flag, name hash, and SID fields from one ring
         element to another.  Since this is called right before a call
         on copsdb in copfil (the only call of this) and since copsdb
         does these copies except for SID (with good reason since it is
         used in other contexts), we could elimiate much of this code.
                                                                        03152
      ARGUMENTS                                                         03153
         ostid - STID - source ring                                     03168
         nstid - STID - destination ring                                03169
      RESULTS                                                           03155
         none                                                           03156
      NON-STANDARD CONTROL                                              03157
         none                                                           03158
      GLOBALS                                                           03159
         none                                                           03160
      EXAMPLE                                                           03161
         none                                                           03162
      %                                                                 03163
   % Declarations %                                                     03164
      LOCAL pgnum, oring, nring;                                        01952
      REF oring, nring;                                                 01953
   pgnum _ lodent(ostid, rngtyp : &oring);                              01954
   frzblk(pgnum, 1);                                                    01955
   ON SIGNAL ELSE frzblk(pgnum, -1); %release frozen page%              01956
   lodent(nstid, rngtyp : &nring);                                      01957
   nring.rnamef _ oring.rnamef;                                         01958
   nring.rtorgin _ oring.rtorgin;                                       01959
   nring.rnameh _ oring.rnameh;                                         01960
   nring.rsid _ oring.rsid;                                             01961
   frzblk(pgnum, -1);                                                   01962
   RETURN;                                                              01963
   END.   %%
```

```
                                                                        03167
%verify%                                                                01965
   (cvertil)   % GB: core NLS Verify File procedure %
   PROCEDURE (fileno, type % => no value %);                            01981
      % Procedure description                                           01982
         FUNCTION                                                       01983
            Verifies that the internal structure of an NLS file is good. 01984
         ARGUMENTS                                                      01985
            fileno - INTEGER - NLs file number of the file              01986
            type - INTEGER - must be (for historical reasons) non-zero  03666
         RESULTS                                                        01987
            none                                                        01988
         NON-STANDARD CONTROL                                           01989
            none                                                        01990
         GLOBALS                                                        01991
            none                                                        01992
         EXAMPLE                                                        01993
            cverfil(stid.stfile, 1)                                     01994
         %                                                              01995
      % Declarations %                                                  01996
      <VERIFY, vfmain> (fileno, type);                                  02006
      RETURN;                                                           02007
      END.   %%
```

FINISH

01999
01971

CHELP

```
< NLS, CHELP.NLS;59, >, 16-NOV-76 01:41 DSM ;;;;
FILE chelp  % L10 <REL-NLS>CHELP.REL %% (L10,) (rel-nls,chelp.rel,) %
                                                                      07213
   % L10 <rel-nls>chelptest %% (L10,) (rel-nls,chelptest.rel,) %      08855
% DECLARATIONS %                                                      07214
   DECLARE giblstd;                                                   08799
   REGISTER r1 = 1;                                                   07215
   REF inpt, conrng, qda, cda, qsw, curstk, qstorblk, qnewstmt,
   hipcmdstk;                                                         07216
   % ********** %                                                    08968
      REF menusw;    % global menu sequence generator -- only ONE %
                                                                      08969
   NUMESS;                                                            08930
      %%                                                             08831
```

```
% Initialization and display %                                          07218
  (qdisp) PROCEDURE (stid);                                            07219
    % (dav) Displays a statement and its menu %                        07220
    LOCAL dummy, endflg;                                               07221
    % ********** %                                                    08859
      REF qsw, qda, menusw;                                            08860
    % set up the user sequence generator address for use by printg.
    The only case in which qsw is non-zero here is on continued menus
    %                                                                  07222
      IF NOT &qsw THEN                                                 07223
        BEGIN                                                          07224
        IF NOT stid THEN RETURN(FALSE);                                07225
        % A new sequence %                                             07226
          &qsw _ openseq(stid, stid, qda.davspec, qda.davspc2,
          qda.dausqcod, qda.dacacode);                                 07227
        % Initialize globals for queryseq %                            07228
          qspcreset(TRUE); % default query viewspecs %                 07229
          qmenucnt _ 0;                                                07230
        qda.dacsp _ stid;                                              07231
        qda.daccnt _ 1;                                                07232
        overscreen _ FALSE;                                            07233
        END                                                            07234
      ELSE IF NOT moremenu THEN                                        07235
        BEGIN                                                          07236
          % This is an error condition; if there is a sequence
          around, should not get in here unless moremenu is TRUE --
          more to be shown.  Close the sequence and return FALSE.
          Clean up the last work area if there is one %                07237
        seqgen(&qsw); % To clean up %                                  07239
        qda.davspec _ qsw.swvspec;                                     07240
        qda.davspc2 _ qsw.swvsp2;                                      07241
        IF nlmode = fulldisplay THEN                                   07242
          dspvsp(qda.davspec, qda.davspc2, 3);                         08822
        closeseq(&qsw := 0);                                           07243
        RETURN( FALSE );                                               07244
        END;                                                           07245
    ON SIGNAL ELSE RETURN(FALSE);                                      07246
    % Display node and menu. %                                        07247
      IF nlmode = fulldisplay THEN                                     07248
        dafrmt(&qda, &qsw : dummy, endflg)                             07249
      ELSE                                                             07250
        BEGIN                                                          07251
        <TSPRT, printg>(&qda, stid, stid, brnchv, &qsw);               07252
        endflg _ TRUE;                                                 07253
        END;                                                           07254
    % Check if finished or end of screen or menu limit reached. %
                                                                       07255
      % ********** %                                                  08861
        IF qsw.swcstid = endfil THEN                                   07256
        ********** %                                                  08862
        IF qsw.swcstid = endfil        % printed everything %         08863
        OR (&menusw AND                                                08971
          getftl(menusw.swcstid))      % printed tail of menu %       09015
        OR inptrf THEN                 % <CTRL-O> %                   08970
          BEGIN                                                        07257
          closeseq(&qsw := 0);                                         07258
```

```
            qda.davspec _ qdavspc;                                  07259
            qda.davspc2 _ qdavs2;                                   07260
            IF hseger THEN                                          07261
               err($"Ran out of space making menus.");              07262
            RETURN(1);                                              07263
            END                                                     07264
         ELSE                                                       07265
            BEGIN                                                   07266
            % Menu limit has been reached or end of screen hit %    07267
               overscreen _ NOT endflg;                             07268
            % so we can go through again %                         08892
               qda.dacsp _ qsw.swstid _ qsw.swcstid;                08895
            % user should be asked whether more desired, get answer %
                                                                    07271
               RETURN(-1);                                          07272
            END;                                                    07273
      END.
                                                                    07274
(qstrinit) PROCEDURE (entstd, topstd);                              07275
      % qstrinit should allocate a large block from dspblk, put its
      address in qstorblk;  this will be used to get smaller blocks.
      xhquit will deallocate this large block if it is non-zero. % 07276
      % Called to initialize the stack area as a storage zone for
      help/query.  Makes use of stgmgt routines to set up blocks in the
      display area block pool.  Also responsible for initializing some
      global cells as well as the context ring %                    07277
         % Allocate string %                                        07278
            &qnewstmt _ getstring( 2000, $dspblk);                  07279
         % Get context ring and initialize. %                       07280
            IF NOT (&qstorblk _ getblk(1777B, $dspblk)) THEN        07281
               err($"System storage problem.  Call ARC programmer");
                                                                    07282
            &qstorblk _ &qstorblk + bhl;                            07283
            makezone(&qstorblk, 70B, 70B, 1777B);                   07284
            IF NOT (&conrng _ getblk(67B, &qstorblk)) THEN          07285
               err($"System storage problem.  Call ARC programmer");
                                                                    07286
            conrng[1] _ &qstorblk; % first word in blocks is the address
            of the zone from which it came. %                       07287
            &conrng _ &conrng + 2; % conrng now points beyond the block
            header and the zone address to the actual stack. %      07288
            confre _ 0;                                             07289
         % Initialize the top and entry point. %                    07290
            % For the top point, perhaps we should have two stids: stid
            of the "introduction" branch for the data base and stid of
            the "directory" branch.  Will not need (should not have?) a
            context stack for the top!  Coming into this procedure,
            topstd should be the stid of the origin of the data base.
            From this node, we may extract links to the "intro" and
            directory branches (which may be the same if desired.)
            CHANGE THIS CODE! %                                     07291
               topcon _ topstd; % first word is stid %              07292
               topcon[1] _ 0; % totcnt, constk, and stkyp fields are 0 %
                                                                    07293
            entcon _ entstd; % first word is stid %                 07294
            entcon[1] _ 0; % totcnt, constk, and stkyp fields are 0 %
```

```
                                                                    07295
    % Initialize global for building entry menu %                   07296
      % The first call on pushent should get a stack if necessary
      and put the type and address in the context ring.  The free
      pointer should be incremented then. %                         07297
      &curstk _ $entcon;                                            07298
      newstk _ 1;                                                   07299
    RETURN; % and then print the entry menu for help %              07300
    END.
                                                                    07301
(helphlp) PROCEDURE;                                                07302
  % (dav) low-level fix-up routine %                                07303
    % save the accumulators %                                       07304
    svac1 _ r1; r1 _ $svacs; !BLT r1, svacse;                       07305
    s _ s + 40000040B;                                              07306
    qagain _ TRUE;                                                  07307
    enthelp( TRUE );                                                07308
    !HRLZI r1, svacs;                                               07309
    !BLT r1, 17B;                                                   07310
    r1 _ svac1;                                                     07311
    !JSYS debrk;                                                    07312
    END.
                                                                    07313
(moreterm) PROCEDURE;                                               07314
  %  Must be at a lower level because of symbol conflicts. Closes
  sequences and resets viewspecs. %                                 07315
    % Clean up the last work area if there is one %                 07316
    seqgen(&qsw); % To clean up %                                   07317
    qda.davspec _ qsw.swvspec;                                      07318
    qda.davspc2 _ qsw.swvsp2;                                       07319
    IF nlmode = fulldisplay THEN dspvsp(qda.davspec, qda.davspc2,
    3);                                                             07320
    closeseq(&qsw := 0);                                            07321
    RETURN;                                                         07322
    END.
                                                                    07323
(clohtil) PROCEDURE (astring);                                      07324
  % Given a tool name, returns a jfn for the help file for that
  tool.  Returns false if there is no help for that tool. %         07325
    LOCAL xptry, hjfn;                                              07326
    LOCAL STRING dirstr[40], hfilnm[100];                           07327
    REF astring;                                                    07328
    IF hdebug THEN *dirstr* _ "<XHELP>" ELSE *dirstr* _ "<HELP>";
                                                                    07329
    xptry _ hjfn _ 0;                                               07330
    *hfilnm* _ *dirstr*, *astring*, ".NLS;" ;                       07331
    ON SIGNAL ELSE                                                  07332
      BEGIN                                                         07333
      *hfilnm* _ "<XPROGRAMS>", *astring*, ".NLS;" ;                07334
      xptry _ TRUE;                                                 07335
      END;                                                          07336
    IF NOT xptry THEN hjfn _ cloafil($hfilnm);                      07337
    ON SIGNAL ELSE RETURN(FALSE);                                   07338
    IF xptry THEN hjfn _ cloafil($hfilnm);                          07339
    RETURN(hjfn);                                                   07340
    END.
```

```
                                                                    07341
(conint) PROCEDURE (stid);                                          07342
   % ********** %                                                  08856
      REF conrng, curstk, qda;                                      08838
   % Initializes context %                                         07343
      % **********                                                 08864
         IF stid # conrng[curindex*rnglnt] THEN                     08867
            BEGIN                                                   08868
            newstk _ TRUE;                                          08869
            IF NOT conrng[confre*rnglnt] THEN                       08870
               %% found a free ring element; bump index and confre %%
                                                                    08887
               BEGIN                                                08872
               curindex _ confre;                                   08873
               IF confre = rngmax THEN confre _ 0                   08874
               ELSE BUMP confre;                                    08875
               END                                                  08876
            ELSE %% no empty ring elements so reuse the oldest %%
                                                                    08877
               BEGIN                                                08878
               curindex _ confre;                                   08879
               %% go free up the stack we are going to use now %%
                                                                    08880
               BUMP confre;                                         08881
               END;                                                 08882
            &curstk _ &conrng + curindex*rnglnt;                    08883
            curstk _ stid;                                          08884
            END                                                     08885
         ELSE newstk _ FALSE;                                       08886
         ********** %                                              08865
         newstk _ TRUE;                                             07346
         IF NOT conrng[confre*rnglnt] THEN                          07347
            % found a free ring element; bump index and confre %
                                                                    08888
            BEGIN                                                   07349
            curindex _ confre;                                      07350
            IF confre >= rngmax THEN confre _ 0                     07351
            ELSE BUMP confre;                                       07352
            END                                                     07353
         ELSE % no empty ring elements so reuse the oldest %        07354
            BEGIN                                                   07355
            curindex _ confre;                                      07356
            freemenu(&conrng + curindex*rnglnt);                    08986
            % go free up the stack we are going to use now %       07357
            IF confre >= rngmax THEN confre _ 0                     08984
            ELSE BUMP confre;                                       08985
            END;                                                    07359
         &curstk _ &conrng + curindex*rnglnt;                       07360
         curstk _ stid;                                             07361
      qda.dacsp _ stid;                                             07364
         % so that includes will have proper file for link parses %
                                                                    08889
      qda.daccnt _ 1;                                               07365
      hlpfileno _ stid.stfile;                                      07366
      % mark the current file not to be closed. %                  07367
         [flntadr(hlpfileno)].flnoclos _ TRUE;                      07368
```

```
        RETURN;                                                    07369
        END.
                                                                   07370
(chkdsp) PROCEDURE (stid);                                         07371
   % displays node at stid %                                       07372
      dismes(0);                                                   07373
      CASE qdisp(stid) OF                                          07374
         = 1: % Everything OK; no more to be shown. %              07375
            BEGIN                                                  07376
            IF hseqer THEN err($"Ran out of space making menus.");
                                                                   07377
            moremenu _ FALSE;                                      07378
            RETURN                                                 07379
            END;                                                   07380
         = -1: % Things OK for now, but must get more interaction
         from user. %                                              07381
            BEGIN                                                  07382
            IF hseqer THEN err($"Ran out of space making menus.");
                                                                   07383
            moremenu _ TRUE;                                       07384
            RETURN                                                 07385
            END;                                                   07386
         ENDCASE % Terrible error. %                               07387
            BEGIN                                                  07388
            moremenu _ FALSE;                                      07389
            err($"Data base portrayal trouble.  Call ARC!");       07390
            END;                                                   07391
      RETURN;                                                      07392
      END.
                                                                   07393
(freemenu) PROCEDURE (conptr);                                     08987
   % given the address of a context node this procedure frees up any
   menus associated with this node%                                08988
   LOCAL menblk, nxtmenblk;                                        08989
   REF conptr,menblk,nxtmenblk;                                    08990
      IF conptr.stktyp THEN &menblk _ conptr.constk                08991
      ELSE RETURN(TRUE);                                           08992
      conptr.stktyp _ conptr.constk _ conptr.totcnt _ 0;           09000
      LOOP                                                         08993
         BEGIN                                                     08994
         &nxtmenblk _ menblk.conlnk;                               08996
         IF NOT freeblk(&menblk-2, menblk[-1]) THEN err($"attempt to
         free invalid storage block");                             08997
         IF &nxtmenblk = -1 THEN RETURN(TRUE) ELSE &menblk _
         &nxtmenblk;                                               08998
         END;                                                      08995
      END.                                                         08999
% Miscellaneous command control %                                 07394
% Search %                                                        07395
(qgetup) PROCEDURE (stid, da, vspstg);                             07396
   % given a stid, display area, and viewspec string, finds the
   source of the current branch.  If it is the origin of a file,
   looks for "source: <link>" to define the source.  Puts viewspecs
   from the link in vspstg. %                                      07397
      LOCAL STRING lnk[100];                                       07398
      LOCAL TEXT POINTER ptr1, ptr2, mkr;                          07399
```

```
      REF vspstg, da;                                                    07400
      IF ( stid.stpsid = origin ) AND (FIND SF(stid) ["source: "]
      ^ptr1 '<  ['>] ^ptr2) THEN                                         07401
         BEGIN                                                           07402
         *lnk* _ ptr1 ptr2;                                              07403
         <wfndit>($ptr1, &vspstg, $mkr, &da);                            07404
         RETURN(mkr);                                                    07405
         END;                                                            07406
      RETURN(getup(stid));                                               07407
      END.
                                                                         07408
(qsearch) PROCEDURE (list, index);                                       07409
   % Given the address of a string containing the node list of an
   item to be found and an index to the context ring element from
   which the search will start, search for the specified node.
   Returns TRUE and the stid of the found node (or FALSE and -1 in
   stid if nothing found). %                                             07410
      LOCAL                                                              07411
         conad,   % address of current context %                         07412
         stid,    % of found item or -1 %                                07413
         nwindex, % of ring or old index if not found %                  07414
         number;                                                         07415
      LOCAL TEXT POINTER ptr, cursptr;                                   07416
      LOCAL STRING vspstg[100], menum[100];                              07417
         % String to be used for getting menu number %                   08819
      REF list, conad;                                                   07418
      IF NOT list.L THEN RETURN(TRUE, qda.dacsp, index);                 07419
      % Initialize for search %                                          07420
         stid _ -1;                                                      07421
         nwindex _ index;                                                07422
         vspstg.L _ fwrd _ 0;  % for bugging, not used in Help %
                                                                         07423
         &conad _                                                        07424
            IF index = entind THEN $entcon                               07425
            ELSE &conrng + index*rnglnt;                                 07426
         glblstd _ 0;                                                    07427
      % see if it starts with a menu number, otherwise call wfndit %
                                                                         07428
         IF NOT <menustart>($menum,&list) THEN                           07429
            BEGIN                                                        07430
            cursptr _ qda.dacsp;  cursptr[1] _ 1;                        07431
            *list* _ '<, *list*, '>;  % so lnkprs can be used %          07432
            FIND SE(*list*) CH ^ptr;                                     07433
            IF NOT<wfndit>($ptr,$vspstg,&list,$cursptr,&qda) THEN
                                                                         07434
               BEGIN                                                     07435
               IF cursptr = qda.dacsp THEN % we assume the whole link
               is bad %                                                  07436
               RETURN(FALSE, -1, index);                                 07437
               % *list* now contains the portion of the address that
               failed but we don't tell the user what failed ? %
                                                                         07438
               END;                                                      07439
            RETURN(TRUE, cursptr, index);                                07440
            END                                                          07441
         ELSE                                                            07442
```

```
            BEGIN                                                   07443
            number _ VALUE($menum);                                 07444
            IF NOT <numfst>(&conad, stid, number, nwindex: stid,
            nwindex) THEN                                           07445
               RETURN (FALSE, -1, nwindex);                         08820
            cursptr _ stid;  cursptr[1] _ 1;                        07446
            <wbrsrh>(&list,$cursptr,&qda);                          07447
            RETURN(TRUE,cursptr,nwindex);                           07448
            END;                                                    07449
      END.
                                                                    07450
(menustart) PROCEDURE (mennum, adr);                                07451
   % Accepts address of a string to be updated and one to be
   evaluated.  Updates the first string to contain just the menu
   number if the second begins with a menu number.  Updates the
   second string with whatever is after the menu number or NULL if
   only non printing characters.  Returns false if it does not begin
   with a menu number. %                                            07452
      LOCAL TEXT POINTER ptr1, ptr2;                                07453
      REF adr, mennum;                                              07454
      mennum.L _ 0;                                                 07455
      IF NOT ( FIND SF(*adr*) ^ptr1 $NP D $D ^ptr2 (NP/ENDCHR) ) OR
      (FIND ptr1 $NP '0) THEN RETURN(FALSE);                        07456
      *mennum* _ ptr1 ptr2;                                         07457
      *adr* _ ptr2 SE(*adr*);                                       07458
      IF FIND SF(*adr*) $NP ENDCHR THEN adr.L _ 0;                  07459
      RETURN(TRUE);                                                 07460
      END.
                                                                    07461
(numfst) PROCEDURE (conad, stid, number, index);                    07462
   % The first item in the list is a menu number.  Count down through
   menu stack "n" items.  It is an error if the number is too high %
                                                                    07463
      LOCAL wrkaddr, nwindex;                                       07464
      LOCAL STRING message[100];                                    07465
      REF conad, wrkaddr;                                           07466
      &conad _ qvalid(&conad, index: nwindex);                      07467
      IF conad.totcnt < number THEN                                 07468
         BEGIN                                                      07469
         *message* _ STRING(number)," is an invalid menu number";
                                                                    07470
         dismes(2,$message); % for the display, we should perhaps put
         message on the screen %                                    07471
         RETURN (FALSE,stid, index);                                07472
         END;                                                       07473
      &wrkaddr _ conad.constk;                                      07474
      LOOP % over menu blocks in this stack %                       07475
         BEGIN                                                      07476
         IF number <= stkmax THEN                                   07477
            BEGIN                                                   07478
            stid _ wrkaddr[number*2];                               07479
            RETURN (TRUE,stid, nwindex);                            07480
            END                                                     07481
         ELSE                                                       07482
            BEGIN                                                   07483
            &wrkaddr _ wrkaddr;                                     07484
```

```
            number _ number - stkmax;                                    07485
            END;                                                         07486
         END;                                                            07487
      END.
                                                                         07488
(conbck) PROCEDURE (index);                                              07489
   % back up one context if possible. otherwise go to etrypoint %
                                                                         07490
   % Check for entrypoint BEFORE callng this procedure !!! %             07491
      CASE index OF                                                      07492
         >0: IF (index _ index-1) = confre THEN index _ entind;          07493
         ENDCASE % =0 %                                                  07494
            index _ IF conrng[rngmax*2] THEN rngmax ELSE entind;
                                                                         07495
      RETURN( index );                                                   07496
      END.
                                                                         07497
(qvalid) PROCEDURE (conad, index);                                       07498
   % Check if this stack has any entries; back up if not %              07499
      REF conad;                                                         07500
      UNTIL conad.constk OR index = entind DO                            07501
         BEGIN                                                           07502
         index _ conbck( index );                                        07503
         &conad _                                                        07504
            IF index = entind THEN $entcon                               07505
            ELSE &conrng + index*rnglnt;                                 07506
         END;                                                            07507
      RETURN(&conad, index );                                            07508
      END.
                                                                         07509
(windit) PROCEDURE (ptr, vspstg, wadr, cursptr, wda);                    07510
   % Given an address to a pointer to a wuc link, an address to a
   viewspec string, an address to the string containing the wuc link
   (wadr), and a pointer to be updated with the new location in the
   file, evaluates the link in relation to its current location, adds
   viewspecs to vspstg and returns true if the address was completely
   successful.  Goes as far as it can and updates cursptr, returns
   false and the portion of the link that failed in wadr if it
   failed. %                                                             07511
      LOCAL STRING filenm[100];                                          07512
      REF wda, ptr, vspstg, wadr, cursptr;                               07513
      ON SIGNAL ELSE                                                     07514
         BEGIN                                                           07515
         IF NOT [sysmsg].L THEN *[sysmsg]* _ "huh";                      07516
         *wadr* _ *[sysmsg]*;                                            07517
         RETURN (FALSE);                                                 07518
         END;                                                            07519
      IF <wprsfile>(&ptr, &vspstg, &wadr, $filenm) THEN                  07520
         RETURN(<wgetfile>($filenm, &wadr, &cursptr, &wda));             07521
            % we don't call wgetadr if a file has been specified
            because since wgetadr continues searching higher levels
            if the first name is not found it would be inconsistant
            with the rule that succeeding elements are always taken
            in relation to the preceeding elements. %                    07522
      IF wadr.L THEN                                                     07523
         RETURN(<wgetadr>($wadr, $cursptr, &wda, &vspstg));              08821
```

```
    RETURN(TRUE);                                              07524
    END. %%                                                    07525
```

```
(wprsfile) PROCEDURE (ptr, vspstg, wadr, filenm);                    07526
   % Parses the link in wadr and returns TRUE if there is a file
   address %                                                         07527
      % declarations %                                              07528
         LOCAL TEXT POINTER ptr1, ptr2;                            07529
         LOCAL ndxcnt, hostno;                                     07530
         LOCAL STRING wstovs[100], wstoad[100];                    07531
         LOCAL STRING hnmono[5], locflnm[40], usrnam[40];          07532
         LOCAL ldstr[35];  % link parsing record %                 07533
      REF ptr, vspstg, wadr, filenm;                               07534
      FIND SE(*wadr*) ^ptr1;                                       07535
      lnkprs($ptr1, $ldstr);                                       07536
      ldstr[lfn] _ ptr.stfile; % because we copied bugged link into
      wadr and called lnkprs with pointer to copied link (to allow
      first word of link to be named).  Why do typed-in links where
      ptr has no stfile work? %                                    07537
      CASE hostno _ lnkpspc( 1, 0, $usrnam, $filenm, $wstoad,
      $wstovs, $ldstr) OF                                          07538
         = lhostn: NULL;  % local host %                           07539
         ENDCASE                                                   07540
            BEGIN                                                  07541
            *wadr* _ "Remote file access not implemented";         07542
            RETURN(FALSE);                                         07543
            END;                                                   07544
      *wadr* _ *wstoad*;                                           07545
      *vspstg* _ *vspstg*, *wstovs*;                               07546
      IF ldstr[fe+1] > ldstr[fs+1] THEN RETURN (TRUE);  % file name %
                                                                   07547
      RETURN (FALSE);                                              07548
      END.
                                                                   07549
(wgetfile) PROCEDURE (filenm, wadr, cursptr, wda);                   07550
   % gets file.  If file not on line, signals with FILE NOT ON LINE
   message.  Finds location in file if any, sets up default
   viewspecs, search specs and returns TRUE.  If it fails, puts bad
   name in wadr. %                                                 07551
      LOCAL TEXT POINTER ptr1, ptr2;                               07552
      LOCAL STRING locadr[100], dumbvs[100];                       07553
      REF wda, wadr, cursptr;                                      07554
      dumbvs.L _ 0;                                                09016
      *locadr* _ "0";                                              09018
      cursptr _ nfstid(filenm, $locadr, $dumbvs, % we dont want vspc
      g % &wda);                                                   07555
         % nfstid returns updated vs, cacode, uscode, but there
         should be no need to get these. It calls caddexp for infile
         address. does not trap any signals %                      07556
      % set up default search type for file (when implemented) %
                                                                   07557
      % set up default viewspecs for file %                        07558
         IF FIND SF(cursptr) ["default view: <:"] ^ptr1 $LD "> ^ptr2
         _ptr2 THEN *defvspc* _ ptr1 ptr2;                         07559
      cursptr[1] _ 1;                                              07560
      IF wadr.L THEN RETURN(wbrsrh(&wadr,$cursptr,&wda));          07561
      RETURN(TRUE);                                                07562
      END.
                                                                   07563
```

```
(wgetadr) PROCEDURE (wadr, cursptr, wda, vspstg);                    07564
   % Finds an address within a multi-file database %                 07565
      LOCAL onlywrd;                                                 07566
      LOCAL TEXT POINTER ptr, ptr1, ptr2;                            07567
      LOCAL STRING stmtnm[120];                                      07568
      % ********** %                                               08857
         REF wadr, cursptr, wda, vspstg;                             07569
      onlywrd _ FALSE;                                               07570
      IF fwrd OR (FIND SF(*wadr*) (L/"@/ % "&/ % "(/")) ^ptr)  THEN
                                                                     07571
         BEGIN                                                       07572
            % starts with statement name not preceeded by space.  If
            you specify the same name as where you already are, your
            current branch will not be searched unless you place a
            space in front of the name.  This limits the search to
            your current branch or a link in your current branch. %
                                                                     08832
         nmdr($ptr,$ptr1,$ptr2);  % Take out name %                  07573
         *stmtnm* _ ptr1 ptr2;                                       07574
         IF FIND ptr2 > $NP ENDCHR THEN onlywrd _ TRUE               07575
         ELSE                                                        07576
            BEGIN                                                    07577
            onlywrd _ FALSE;                                         07578
            *wadr* _ ptr2 SE(*wadr*);                                07579
            END;                                                     07580
         IF NOT <wstrtnm>($stmtnm, &cursptr, &wda, &vspstg, onlywrd)
                                                                     08800
         THEN BEGIN                                                  08801
         % **********                                               08840
            IF cursptr.stfile = hlpfileno THEN                       08802
            ********** %                                            08843
            IF cursptr.stfile = helpfirstfile THEN                   08842
               BEGIN    % Already tried the initial file %           08803
               *wadr* _ *stmtnm*;                                    08804
               RETURN(FALSE);                                        08805
               END;                                                  08806
         % **********                                               08844
            cursptr.stfile _ hlpfileno;                              08807
               %% Start with the initial file %%                    08808
            ********** %                                            08845
            FIND cursptr ^ptr;    % save cursptr %                 08978
            cursptr.stfile _ helpfirstfile;                          08847
            cursptr.stpsid _ origin;                                 08858
               % Start at the origin of the initial file %          08848
         IF NOT <wstrtnm>($stmtnm, &cursptr, &wda, &vspstg,          08809
            onlywrd) THEN                                            08810
               % This may search some (index) files that have
               already been searched.  Oh well. %                    08811
            BEGIN                                                    08812
            % ********** %                                          08975
               FIND ptr ^cursptr;    % restore cursptr %           08976
            *wadr* _ *stmtnm*;                                       08813
            RETURN(FALSE);                                           08814
            END;                                                     08815
         END;                                                        08816
      END;                                                           07587
```

```
    IF NOT onlywrd THEN    % get the rest of the address %     07588
       % **********                                           08849
          RETURN(<wbrsrh>(&wadr,$cursptr,&wda))               07589
          ********** %                                       08850
          RETURN(<wbrsrh>(&wadr,&cursptr,&wda))               08852
    ELSE RETURN(TRUE);                                        07590
    END.
                                                              07591
(wstrtnm) PROCEDURE (stmtnm, cursptr, wda, vspstg, onlywrd);  07592
   % finds the first name in a multi-file database and returns true.
   Returns false if it can't find it.  Uses vspstg to check for view
   if necessary when the same name has been requested twice in a row.
   %                                                          07593
      % declarations and REFs %                              07594
         LOCAL TEXT POINTER movmkr, ptr1, ptr2;               07595
         LOCAL STRING savenm[100];                            07596
         REF vspstg, stmtnm, cursptr, wda;                    07597
      % initializations %                                    07598
         movmkr _ cursptr;                                    07599
         movmkr[1] _ cursptr[1];                              07600
      IF cursptr # glblstd THEN % we have not been called from wbranm
      so do the branch search %                               07601
         BEGIN                                                07602
         IF fwrd OR <samename>($stmtnm,&cursptr) THEN %  Check if the
         view is the same.  If the first word of a statement was
         bugged or the current statement's name has been requested,
         then we assume the user wants a more general concept at a
         higher level unless a different view has been specified. %
                                                              07603
            BEGIN                                             07604
            % Check if we are at the same statement %         07605
            IF (cursptr = wda.dacsp) AND (wda.daccnt = 1) THEN 07606
               BEGIN                                          07607
               IF NOT ( inhelp OR athelp ) AND onlywrd AND NOT
               <samevw>(&vspstg, &wda) THEN RETURN(TRUE);     07608
                  % If there were something following the name (NOT
                  onlywrd), then since in every case you can find the
                  address in relation to the current location without
                  typing the name of the current location we assume
                  you were looking for a statement with the same name
                  elsewhere and so skip this. %               07609
               END                                            07610
            ELSE IF fwrd THEN % take the user there even if it's not
            named % RETURN(TRUE);                             07611
               % if fwrd and not onlywrd, then we don't go searching
               higher level %                                 07612
            END                                               07613
         ELSE % search current branch %                       07614
            BEGIN                                             07615
            *savenm* _ *stmtnm*;                              07616
            IF <wbranm>($stmtnm,$movmkr,&wda) THEN            07617
               BEGIN                                          07618
               cursptr _ movmkr;                              07619
               RETURN(TRUE);                                  07620
               END;                                           07621
            IF inptrf THEN RETURN(FALSE);                     07622
```

```
            movmkr _ cursptr;  movmkr[1] _ 1;  % movmkr = endfil %
                                                                    07623
            END;                                                    07624
         END                                                        07625
      ELSE glblstd _ 0;                                             07626
      % Get search type %                                          07627
         % Before changing the search type to alphabetic in the top
         index, this should be changed to use the search type defined
         in the origin statement.  This might be the top index.  We
         should either set a global defining the searchtype whenever
         we enter a new file or we enter new command and the file has
         changed, or we should look in the origin statement at this
         point. >  See <wgetfile> %                                 07628
      % search this file %                                          07629
         IF ( movmkr _ lookup ($movmkr, $stmtnm, nametyp) ) # endfil
         THEN % we have found it %                                  07630
            IF movmkr # wda.dacsp THEN % it is not the same as our
            current location %                                      07631
               BEGIN                                                07632
               cursptr _ movmkr; cursptr[1] _ 1;                    07633
               RETURN(TRUE);                                        07634
               END                                                  07635
            ELSE IF NOT ( inhelp OR athelp ) THEN  % check to see if
            the view is the same %                                  07636
               IF NOT <samevw>(&vspstg, &wda) THEN                  07637
                  BEGIN                                             07638
                  cursptr _ movmkr; cursptr[1] _ 1;                 07639
                  RETURN(TRUE);                                     07640
                  END;                                              07641
      % search file indexes %                                       07642
         RETURN (<srchfndx>(&cursptr, $movmkr, $stmtnm, &wda));     07643
      END.
                                                                    07644
(samename) PROCEDURE (stmtnm, cursptr);                             07645
   % Compares a statement name with the one in the statement at
   cursptr.  Returns true if same else false. %                     07646
      LOCAL spcnmhash;                                              07647
      LOCAL STRING locstr[100];                                     07648
      REF stmtnm, cursptr;                                          07649
      astruc(&stmtnm);                                              07650
      spcnmhash _ <hash>(&stmtnm);                                  07651
      IF NOT <getnmf>(cursptr) OR <getnam>(cursptr) # spcnmhash THEN
                                                                    07652
         RETURN(FALSE);                                             07653
      CCPOS SF(cursptr);                                            07654
      xtrnam($locstr, $swork, -1, 0);                               07655
      RETURN( IF *stmtnm* = *locstr* THEN TRUE ELSE FALSE ) ;       07656
      END.
                                                                    07657
(samevw) PROCEDURE (vspstg, wda);                                   07658
   % returns true if the viewspecs evaluated in vspstg are the same
   as those in wda false otherwise %                                07659
      LOCAL same, vs1, vs2, cacode, usedgen;                        07660
      REF vspstg, wda;                                              07661
      *vspstg* _ *defvspc*, *vspstg*;                               07662
      % save viewspecs %                                           07663
```

```
      vs1 _ wda.davspec; vs2 _ wda.davspc2;                              07664
      cacode _ wda.dacacode; usedgen _ wda.dausqcod;                     07665
   IF vspstg.L # 0 THEN % set up viewspecs %                             07666
      BEGIN                                                              07667
      % if you typed in the name and then hit CA, there will be no
      viewspecs, therefore, no matter what the view, you will go
      to a higher level search.  The reasoning is if you wanted a
      map view of the current location, you would not have typed
      the name, you wuld have just hit CA %                              07668
      % have to use davspec for feedlt %                                 07669
         wda.davspec _ cspvs; wda.davspc2 _ cspvs[1];                    07670
         wda.dacacode _ cspcacode; wda.dausqcod _ cspusqcod;             07671
      feedlt(&wda,&vspstg); % evaluate the viewspecs %                   07672
      END;                                                               07673
   IF (( cspvs # wda.davspec ) OR ( cspvs[1] # wda.davspc2 )) THEN
   same _ FALSE ELSE same _ TRUE ;                                       07674
   % replace old viewspecs to fool howformat %                          07675
      wda.davspec _ vs1; wda.davspc2 _ vs2;                              07676
      wda.dacacode _ cacode; wda.dausqcod _ usedgen;                     07677
   RETURN (same) ;                                                       07678
   END.
                                                                         07679
(srchfndx) PROCEDURE (cursptr, movmkr, stmtnm, wda);                     07680
   % Follows index links to other help files %                          07681
      LOCAL hostno, ndxcnt;                                              07682
      LOCAL TEXT POINTER savmkr, ptr1;                                   07683
      LOCAL STRING usrnam[40], filenm[80], adr[100], vspcs[100];
                                                                         07684
      LOCAL ldstr[35];                                                   07685
      REF wda, cursptr, movmkr, stmtnm;                                  07686
      IF inptrf THEN RETURN(FALSE);    % the user typed ctrl-0 %
                                                                         07687
      ndxcnt _ 0;                                                        07688
      movmkr    _ savmkr    _ cursptr;                                   07689
      movmkr[1] _ savmkr[1] _ 1;       % changed to endfile %            08979
      LOOP    % see if this file has it's own index %                    07690
         BEGIN                                                           07691
         movmkr.stpsid _ origin;                                         07692
         % ********** %                                                 08982
            movmkr _ getsub(movmkr);                                     08980
               % Look in statement 1 for an index denoted by the
               string "Index in <...>" %                                 08981
         IF NOT FIND SF(movmkr) ["ndex in <"] [">] ^ptr1 _ptr1 THEN
                                                                         07693
            EXIT LOOP;                                                   08983
         dismes(1,$"searching index");                                   07694
         % find the file %                                               07695
            ON SIGNAL ELSE                                               07696
               BEGIN                                                     07697
               IF inptrf THEN RETURN(FALSE);                             07698
               dismes(1,$"Bad index link");                              07699
               RETURN(FALSE);                                            07700
               END;                                                      07701
            lnkprs($ptr1, $ldstr);                                       07702
            hostno _ lnkpspc(1, $ptr1, $usrnam, $filenm, $adr,
            $vspcs, $ldstr);                                             07703
```

```
            IF hostno # lhostn THEN err($"Remote site indexes
            currently not supported.");                          07704
         movmkr _ nfstid($filenm, $adr, $vspcs, &wda: movmkr[1]);
                                                                 07705
         ON SIGNAL ELSE;                                         07706
         IF movmkr.stfile = savmkr.stfile THEN EXIT LOOP; % we've
         already looked here %                                   07707
       % find the name %                                        07708
         savmkr _ movmkr;                                        07709
         IF inptrf THEN RETURN(FALSE);  % the user typed ctrl-0 %
                                                                 07710
         *adr* _ *stmtnm*;  % lookup says it modifies string %
                                                                 07711
         movmkr _ lookup($movmkr, $stmtnm, nametyp);             07712
         *stmtnm* _ *adr*;  % lookup says it modifies string %
                                                                 07713
         IF movmkr # endfil THEN                                 07714
            BEGIN                                                07715
            cursptr _ movmkr; cursptr[1] _ 1;                    07716
            RETURN(TRUE);                                        07717
            END;                                                 07718
         IF inptrf THEN RETURN(FALSE);  % the user typed ctrl-0 %
                                                                 07719
       movmkr _ savmkr;                                          07720
       BUMP ndxcnt;                                              07721
       IF ndxcnt > 10 THEN                                       07722
          BEGIN                                                  07723
          typeas($"reached limit of 10 index files
          ");                                                    07724
          EXIT LOOP;                                             07725
          END;                                                   07726
       END;  % repeat loop %                                     07727
    RETURN(FALSE);                                               07728
    END.

                                                                 07729
(wbrsrh) PROCEDURE (wadr, cursptr, wda);     % <,:b> %           07730
   % ? %                                                        07731
    LOCAL TEXT POINTER ptl1, wnmptr, movmkr, ptr1, ptr2;         07732
    LOCAL STRING stmtnm[100], vspstg[100];                       07733
    REF wda, wadr, cursptr;                                      07734
    FIND SF(*wadr*) ^ptr2;                                       07735
    LOOP % <,:b> to take the user as far as we can after the first
    name %                                                       07736
       BEGIN                                                     07737
       IF FIND ptr2 > $NP ENDCHR THEN RETURN(TRUE);              07738
       movmkr _ cursptr; movmkr[1] _ 1;                          07739
       IF FIND ptr2 > SP $NP (L/'@) ^ptr1 _ptr1 $(LD/'@/''/'-)
       ^ptr2 THEN % there is a stacked name, %                   07740
          BEGIN                                                  07741
          *stmtnm* _ ptr1 ptr2;                                  07742
          IF NOT wbranm($stmtnm, $movmkr, &wda) THEN             07743
             BEGIN                                               07744
             IF getsub(cursptr) # cursptr THEN *wadr* _ *stmtnm*,
             "?  You can pick one and try again"                 07745
             ELSE *wadr* _ *stmtnm*;                             07746
             RETURN(FALSE);                                      07747
```

```
      END;                                                    07748
   cursptr _ movmkr;                                          07749
   IF inptrf THEN                                             07750
      BEGIN                                                   07751
      *wadr* _ *stmtnm*;                                      07752
      RETURN(FALSE);  % the user typed ctrl-0 %               07753
      END;                                                    07754
   END                                                        07755
ELSE                                                          07756
   BEGIN % set up content searches, link searches, and
   branch searches for caddexp (change caddexp to do this
   stuff) %                                                   07757
   IF FIND ptr2 > $NP ENDCHR THEN RETURN(TRUE);               07758
   IF FIND ptr2 > $NP '" ^ptr1 THEN                           07759
      BEGIN                                                   07760
      IF FIND ptr1 > [^"] ('=/) $LD ^ptr2 _ptr1 THEN          07761
         BEGIN                                                07762
         ON SIGNAL                                            07763
            = gaderr:                                         07764
               BEGIN                                          07765
               IF [sysmsg].L # 0 THEN *wadr* _ *[sysmsg]*;
                                                              07766
               RETURN(FALSE);                                 07767
               END;                                           07768
            ELSE;                                             07769
         caddexp($ptr1, $ptr2, &wda, $movmkr);                07770
         ON SIGNAL ELSE;                                      07771
         IF movmkr # endfile THEN cursptr _ movmkr;           07772
         cursptr[1] _ 1;                                      07773
         END                                                  07774
      ELSE *wadr* _ ptr2 SE(*wadr*), '"; % add the quote %
                                                              07775
      REPEAT LOOP;                                            07776
      END;                                                    07777
   FIND ptr2 ^wnmptr;                                         07778
   LOOP                                                       07779
      IF FIND wnmptr > [SP] ^ptr1 _ptr1 (L/'@) ^ptl1 _ptl1
      $(LD/'-/''/'@) ^wnmptr THEN ST ptr1 ptl1 _ " !"         07780
      ELSE                                                    07781
         BEGIN                                                07782
         FIND ptr2 ^wnmptr;                                   07783
         EXIT LOOP;                                           07784
         END;                                                 07785
   LOOP                                                       07786
      IF FIND wnmptr > [D] ^ptr1 _ptr1 $D ^ptl1 '. ^wnmptr
      (ENDCHR/SP) THEN ST ptr1 wnmptr _ '", '<, '", '=, ptr1
      ptl1, "cls" ELSE EXIT LOOP;                             07787
   FIND ptr2 $NP ^ptr1 SE(*wadr*) $NP ^ptr2;                  07788
   IF ptr1[1] < ptr2[1] THEN % call caddexp %                 07789
      BEGIN                                                   07790
      ON SIGNAL = gaderr: GOTO wcaderr; ELSE;                 07791
      caddexp($ptr1, $ptr2, &wda, $movmkr);                   07792
      IF movmkr # endfile THEN                                07793
         BEGIN                                                07794
         cursptr _ movmkr;                                    07795
         cursptr[1] _ movmkr[1];                              07796
```

```
                END;                                                    07797
              END;                                                      07798
            RETURN(TRUE);                                               07799
            END;                                                        07800
        END; % repeat loop %                                            07801
    (wcaderr):                                                          07802
        IF [sysmsg].L # 0 THEN *wadr* _ *[sysmsg]*;                     07803
        RETURN(FALSE);                                                  07804
    END. %%                                                             07805
```

```
(wbranm) PROCEDURE (stmtnm, cursptr, wda);   % <,:b> %                  07806
   % If not in branch, look for a link.  If the link is not
   completely accurate, forget it.  To search a link in every
   statement in substructure is potentially too time consuming. %
                                                                        07807
      REF wda, stmtnm, cursptr;                                         07808
      LOCAL cnnt;                                                       07809
      LOCAL STRING wadrsave[150], vspstg[100];                          07810
      LOCAL TEXT POINTER movmkr, ptr1, ptr2;                            07811
      cnnt _ 0;                                                         07812
      LOOP                                                              07813
         BEGIN                                                          07814
         IF inptrf THEN RETURN(FALSE);  % the user typed ctrl-0 %
                                                                        07815
         movmkr _ cursptr; movmkr[1] _ 1;                               07816
         movmkr _ lookup($movmkr, $stmtnm, (IF cursptr.stpsid =
         origin THEN nametype ELSE braname ));                          07817
         IF movmkr # endfil THEN  % we found it %                       07818
            BEGIN                                                       07819
            cursptr _ movmkr;                                           07820
            RETURN(TRUE);                                               07821
            END;                                                        07822
         % see if there is a link to somewhere else %                   07823
            IF inptrf THEN RETURN(FALSE);  % the user typed ctrl-0 %
                                                                        07824
            movmkr _ glblstd _ cursptr;                                 07825
            IF NOT ( FIND SF(movmkr) ['>] ^ptr2 _ptr2 < ['<] ^ptr1 (
            $NP "##"/EOL/CR  LF ) ) OR ( FIND ptr1 > CH ('',/) $NP
            (':/'>) ) THEN RETURN(FALSE) ;                              07826
         % take it %                                                    07827
            *wadrsave* _ ptr1 ptr2, '>;                                 07828
            IF NOT wfndit($ptr2,$vspstg,$wadrsave,&cursptr,&wda) THEN
            RETURN(FALSE);                                              07829
         IF cursptr = movmkr THEN RETURN(FALSE);                        07830
         movmkr _ cursptr;                                              07831
         IF cnnt > 10 THEN                                              07832
            BEGIN                                                       07833
            <err>($"looping link?  ");                                  07834
            RETURN(FALSE);                                              07835
            END;                                                        07836
         BUMP cnnt;                                                     07837
         END;                                                           07838
      END. %%                                                           07839
```

```
% Sequence generator(s) for printg %                                    07840
  (queryseq) PROCEDURE (sw, phase);                                     07841
    % query sequence generator program %                               07842
    % This controls the printout or display of a QUERY branch when a
    SHOW function is requested.  It handles INCLUDES, Q-specs, and
    menu generation. %                                                  07843
      LOCAL saveview;                                                   07844
      REF sw;                                                           07845
      CASE phase OF                                                     07846
        = sqopn: % OPEN %                                               07847
          RETURN;                                                       07848
        = sqgnxt: % NEXT (below) %                                      07849
          NULL;                                                         07850
        = sqcls: % CLOSE %                                              07851
          RETURN;                                                       07852
        ENDCASE err($"Query System Failure");                           07853
      ON SIGNAL ELSE                                                    07854
        % Some awful failure; send back EOF with error flag set. %
                                                                        07856
        BEGIN                                                           07855
        hseger _ TRUE;    % Error message will be wrong, but... %
                                                                        07857
        sw.swcstid _ sw.swstid _ endfil;                                07858
        sport(&sw);                                                     07859
          % don't want to be called again.  Finished with a query
          branch %                                                      08823
        END;                                                            07860
      qnewstmt.L _ 0;                                                   07861
      saveview _ sw.swvspec;                                            07862
      qstmt(&sw,                                                        07863
        0,      % no previous sequence %                               08826
        TRUE,  % do process the subplex of this statement, if any %
                                                                        08824
        $"");  % process addressed node top statement and build a
        menu for any substructure %                                     08825
      sw.swvspec _ saveview;                                            07864
      % send back EOF because we are done %                            07865
        sw.swcstid _ sw.swstid _ endfil;                                07866
        sport(&sw);                                                     07867
          % don't want to be called again.  Finished with a query
          branch %                                                      08827
      END.
                                                                        07868
  (qstmt) PROCEDURE (sw, prevsw, topflag, appendstr);                   07869
    % process one QUERY source statement %                             07870
    % Examine "current" data base source statement, executing any
    links and performing INCLUDES wherever context and Q-specs
    dictate.  Trigger menu generation only if topflag=TRUE.  ELSE
    ignore substructure to this statement. %                            07871
    % If topflag = TRUE, assumed to be top addressed node, i.e. an
    asterisk as the first character must be stripped off %              07872
    % qstmt will either be called (1) by queryseq to process the
    branch being SHOWN (addressed) (prevsw=0), or (2) by qmenu to
    process a single statement in the plex being menued (uncolumnated
    only) (prevsw#0) %                                                  07873
      LOCAL                                                             07874
```

```
    code,   % Type of statement: menu, non-menu, etc. %     07875
    truncate,  % Do not execute links beyond the frist line if
    TRUE. %                                                 07876
    sendsw,   % temp for sequence work area address %       07877
    nextchr,  % count of next character to be scanned. %    07878
    dirflg; % TRUE = q-directive found %                    07879
  LOCAL TEXT POINTER                                        07880
    oldpos,  % front of this segment %                      07881
    endnew;   % end of this segment %                      07882
  REF sw, prevsw, sendsw, appendstr; % could be top node or a
  linked-to INCLUDED node %                                 07883
  % general initialization %                                07884
    oldpos _ endnew _ sw.swcstid;                           07885
    endnew[1] _ oldpos[1] _ CASE code _ tstmenu(oldpos) OF  07886
      = 3, = 4: 2; % Comment character to be peeled off %   07887
      ENDCASE 1; % Logical front of this statement %        07888
    truncate _ NOT topflag AND code IN [1,2];               07889
    &sendsw _ IF &prevsw THEN &prevsw ELSE &sw;             07891
  % Handle contents of this statement.  Optimize case where no
  directives occur.  No FIND statements used until queryspecs
  encountered. %                                            07892
    LOOP                                                    07893
      BEGIN % keep processing through this stmt.  If this is a
      top level node, process  until segments of text are sent
      and all link list INCLUDES have been taken.  If this is a
      menu item, gather data in global string for single send
      when includes have been taken; avoid CRs generated by
      sends. %                                              07894
      dirflg _ qdirparse ($oldpos, $endnew : nextchr);      07895
      % append and send previous segment if this is top level
      node; otherwise, just append %                        07896
        dirflg _ qappender($qnewstmt, &appendstr, $oldpos,
        $endnew, truncate, dirflg, sendsw.swvspec.vstrnc);
                                                            07897
          % If this is a menu node, only append first 72
          characters or one line; change dirflg to reflect
          limit reached to exit loop. %                     07898
        IF topflag OR NOT dirflg THEN % send out now. %     07899
          qsender($qnewstmt, &sendsw, topflag);             07900
        IF NOT dirflg THEN EXIT LOOP;                       07901
      % advance pointers for include %                       07902
        oldpos[1] _ endnew[1] := nextchr;                   07903
      % do link list found by qdirparse - whether called with a
      menu statement or a main node %                       07904
        qinclude (&sendsw, $qnewstmt, $oldpos, $endnew,
        topflag);                                           07905
          % qinclude calls qstrip to process included stmt,
          Qstrip sends the stmt out without directives %
                                                            07906
      % advance pointers for next scan %                     07907
        oldpos[1] _ endnew[1] _ nextchr;                    07908
      END; % of text/include loop %                         07909
    IF topflag THEN % we are now processing the TOP statement
    (called from queryseq) and about to do MAIN subplex %   07910
      qmenu (sw.swcstid, sw.swlbstid, &sendsw); % do menu as
      requested %                                           07911
```

```
      RETURN;                                                    07912
      END.   % normal exit %
                                                                 07913
(qsender) PROCEDURE (string, workarea, topflag);                 07914
   % sends the contents of string out to sequence workarea with level
   controlled by topflag.  Resets contents of string after send. %
                                                                 07915
   % It avoids sending a null string %                          07916
      LOCAL savlvl;                                              07917
      REF string, workarea;                                      07918
      IF string.L THEN                                           07919
        BEGIN                                                    07920
        savlvl _ workarea.swclvl := workarea.swslvl + (NOT topflag);
                                                                 07921
        send(&workarea, &string);                                07923
        workarea.swclvl _ savlvl;                                07924
        string.L _ 0;                                            07925
        END;                                                     07926
      RETURN;                                                    07927
      END.
                                                                 07928
(qappender) PROCEDURE (string, appendstr, front, end, truncate,
dirflg, lines);                                                  07929
   % Appends contents of appendstr to front of string; puts contents
   delimited by front and end at the end of string.  (Does not try to
   append sppendstr if NULL; does not try to append front through end
   if front is zero.)  Does not reset original contents of the
   string. %                                                     07930
      LOCAL length;                                              07931
      LOCAL TEXT POINTER tp;                                     07932
      REF string, appendstr, front, end;                         07933
      IF appendstr.L THEN                                        07934
         BEGIN                                                   07935
         *string* _ *appendstr*, *string*;                       07936
         appendstr.L _ 0;                                        07937
         END;                                                    07938
      IF front THEN *string* _ *string*, front end;              07939
      IF truncate  THEN                                          07940
         BEGIN                                                   07941
         IF (length _ lines*72) < string.L THEN                  07942
            BEGIN                                                07943
            string.L _ length;                                   07944
            dirflg _ FALSE;                                      07945
            END;                                                 07946
         IF FIND SF(*string*) [EOL ^tp _tp/CR LF ^tp _2tp] THEN  07947
            BEGIN                                                07948
            *string*  _ SF(*string*) tp;                         07949
            dirflg _ FALSE;                                      07950
            END;                                                 07951
         END;                                                    07952
      RETURN(dirflg);                                            07953
      END.
                                                                 07954
(qdirparse) PROCEDURE (start, end);                              07955
   % parse query directives %                                    07956
   % Given the address of a text pointer to the current position in
```

```
   the statement (which will NOT be changed by this procedure), the
   address of a text pointer which may be changed to point to the
   beginning of a qspec block or the end of the statement if no block
   exists, qdirparse locates the query directive string, if any, sets
   end, and returns TRUE and the count character of the next
   character to be scanned.                                          07957
   If no directives are found, it returns FALSE and the count of the
   end of the statement.                                             07958
   It attempts to do the scan quickly to optimize case where no
   directives exist. %                                               07959
      LOCAL                                                          07960
         count,                                                      07961
         drswork[7]; % string work area for READC. %                 07962
      REF start, end;                                                07963
      % Initialize the string work area. %                          07964
         drswork _ start;                                            07965
         drswork[1] _ start[1];                                      07966
         fechc1(forward, $drswork);                                  07967
         count _ 0;                                                  07968
      % Find the first. %                                            07969
         CASE READC($drswork) OF                                     07970
            = '#:                                                    07971
               BEGIN                                                 07972
               IF NOT count THEN                                     07973
                  BEGIN                                              07974
                  IF READC($drswork) # '# THEN REPEAT CASE;          07975
                  end[1] _ drswork[1] - 2;                           07976
                  BUMP count;                                        07977
                  REPEAT CASE;                                       07978
                  END                                                07979
               ELSE                                                  07980
                  BEGIN                                              07981
                  IF READC($drswork) # '# THEN REPEAT CASE;          07982
                  RETURN( TRUE, drswork[1]);                         07983
                  END;                                               07984
               END;                                                  07985
            =ENDCHR:                                                 07986
               BEGIN                                                 07987
               RETURN( FALSE, end[1] _ drswork[1]);                  07988
               END;                                                  07989
            ENDCASE REPEAT CASE;                                     07990
      END.
                                                                     07991
(qinclude) PROCEDURE (sw, string, qlkleft, qlkright, topflag);       07992
   % process query link list (includes) %                            07993
   % Process from global qlkleft to qlkright, obeying user's
   viewspecs in links and the qspecs directives.  Each link in the
   list is executed, and a new sequence is opened at the new
   location. User viewspecs, if any, predominate over query defaults.
                                                                     07994
   The linked-to statement is included without its statement name and
   without any directives ("##....##") it may contain.  Qspecs govern
   the inclusion of the substructure as menu.                        07995
   Before leaving, the file and new sequence are closed.             07996
      File not closed for now; need file management code to  take
      care of stids if files must be closed.                         07997
```

```
PARAMETERS:  address of sequence work area (main), flag for
including name on linked-to statement(s).  If topflag is true, we
are processig the topnode or links in the top node.  This affects
whether or not we will accept new qspecs.  Generally no if false--
formatting based on the top node. %                                  07998
% NOTE: assuming format of "##[..](..)(..)[..](..)etc##  If a
queryspecs setting occurs at the end of the string, it will
pertain to the substructure of this node (if applicable) since it
stands alone.  This, of course, will be erased if another list
occurs before substructure of this node processed.  To set qspecs
to affect the sub of the node begin addressed, use a [..] last in
the list (or only element). %                                        07999
   LOCAL                                                             08000
      nofind,   % to avoid looping: TRUE if neither item located %
                                                                     08001
      stid;    % for new file/stmt %                                08002
   LOCAL TEXT POINTER                                                08003
      curpos,    % current position in string being processed %
                                                                     08004
      oldpos,    % temp for position %                              08005
      startspec,   % start of qspec or link %                       08006
      endspec;     % end of qspec or link %                         08007
   LOCAL STRING qstn[200]; % statement number %                      08008
   REF                                                               08009
      sw,                                                            08010
      string,                                                        08011
      qlkleft,                                                       08012
      qlkright;                                                      08013
   ON SIGNAL ELSE RETURN; % currently, this is a way out - lnkspec
   will complain when it finds no more links %                       08014
   % initialization %                                               08015
      IF qlkleft[1] = qlkright[1] THEN RETURN;                       08016
      stid _ curpos _ oldpos _ qlkleft;  % in case no link, just
      viewspecs %                                                    08017
      curpos[1] _ endspec[1] _ qlkleft[1];                           08018
      % strip garbage off end of link list %                        08019
         FIND qlkright < 2CH $NP ^qlkright;                          08020
   % run through list, getting [..] first, then (...)(...) etc. %
                                                                     08021
   LOOP                                                              08022
      BEGIN                                                          08023
      % process QSPECS   %                                          08024
         nofind _ FALSE;                                             08025
         IF FIND curpos > 2CH $NP ^endspec "[ ^startspec  $(LD/"=)
         % NOTE:  really ought to use FS and let anything go by,
         just trapping ] for this to handle errors in the best way
         % "] < CH ^endspec THEN                                     08026
            % found QSPECS %                                         08027
            BEGIN                                                    08028
            qsparse ($startspec, $endspec, topflag);                 08029
            curpos[1] _ endspec[1] + 1; % next char to process %
                                                                     08030
            END                                                      08031
         ELSE BUMP nofind;                                           08032
         IF (oldpos[1] _ curpos[1]) >= qlkright[1] THEN EXIT LOOP;
                                                                     08033
```

```
      % process next link %                                          08034
         % Parse link and get stid. %                                 08035
            qlnkspec($curpos, $stno, $stn2 % filename % ,$qstn %
            number % , $num % vspecs % ); % this will go to err if
            no more links %                                          08036
            IF stn2.L % filename exists % THEN                       08037
               stid _ nfstid($stn2 % filename % , $qstn % stmt no
               % , $num % vspecs % ) % open the file %               08038
            ELSE stid _ ofstid( &qda, $qstn);                        08039
         % concatenate user string with ours (esb), ours first;
         change viewspecs only if not being menued.  Otherwise
         must only put out one line! %                               08040
            IF topflag THEN *num* _ "esb", *num*                     08041
            ELSE *num* _ "te"; % one line, one level in menu %
                                                                     08042
            feedsw(&sw, $num % vspecs % ); % put new viewspecs in
            the sequence work area %                                 08043
         % remove directives, send out %                              08044
            qstrip(stid, &sw % destination of qstrip's output % ,
            topflag, &string);                                       08045
         IF topflag AND qspecs.querinc THEN                          08046
            % will not be TRUE if NOT topflag (see qsparse) %
                                                                     08917
            qmenu(stid, endfil, &sw);                                08048
               % do a menu if user-specified %                      08918
      % test for done %                                            08050
         CASE curpos[1] OF                                           08051
            = oldpos[1]:    % no more links in statement %          08052
               EXIT LOOP;                                            08053
            >= qlkright[1]: % end of section of stmt to process %
                                                                     08054
               EXIT LOOP;                                            08055
            ENDCASE nofind _ FALSE; % we found link %                08056
         BUMP curpos[1]; % advance to next character %               08057
      IF nofind THEN EXIT; % didn't find [...] or (...) - DB
      error.  Just ignore %                                          08058
      END;    % of qspecs/links processing loop %                   08059
   RETURN;                                                           08060
   END.
                                                                     08061
(qsparse) PROCEDURE (qspstrt, end, topflag);                         08062
   % set global queryspecs %                                        08063
   % interpret string between qspstrt and end and set proper flags in
   global, qspecs - WARNING to data base builder - this is a
   hard-nosed son-of-a-bitch routine. %                              08064
      LOCAL                                                          08065
         char,   % next char to process %                           08066
         number,                                                     08067
         term, % number of characters to scan %                      08068
         qspwork[7]; % string work area %                            08069
      LOCAL STRING nmstr[10];                                        08070
      REF qspstrt, end;                                              08071
      % initialize string work area. %                              08072
         qspwork _ qspstrt;                                          08073
         qspwork[1] _ qspstrt[1];                                    08074
         fechc1( forward,  $qspwork);                                08075
```

```
      term _ end[1] - qspstrt[1];                                    08076
    number _ FALSE;                                                  08077
    qspcreset(topflag); % set default values for qspecs %            08078
    FOR term DOWN UNTIL <= 0 DO                                      08079
      BEGIN                                                          08080
      CASE char _ READC($qspwork) OF % get next char of statement
      %                                                              08081
        ='c, ='C:                                                    08082
          IF topflag THEN qspecs.quercol _ TRUE;                     08083
        ='m, ='M: % Do not include substructure; main node only %
                                                                     08084
          IF topflag THEN qspecs.querinc _ FALSE;                    08085
        ='p, ='P:                                                    08086
          qspecs.querprt _ TRUE;                                     08087
        ='n, ='N:                                                    08088
          number _ TRUE;                                             08089
        ='=:                                                         08090
          IF NOT number:=number+1 THEN EXIT LOOP;                    08091
        = D: % digit %                                               08092
          BEGIN                                                      08093
          IF number # 2 THEN EXIT LOOP;                              08094
          IF nmstr.L = nmstr.M THEN EXIT LOOP; % avoid overflow
          %                                                          08095
          *nmstr* _ *nmstr*, char;                                   08096
          END;                                                       08097
        = NP: REPEAT LOOP;                                           08098
        ENDCASE EXIT LOOP;                                           08099
      END; % LOOP %                                                  08100
    IF topflag AND nmstr.L THEN                                      08101
      qspecs.queropts _ VALUE($nmstr); % may truncate value! %
                                                                     08102
    RETURN;                                                          08103
    END.
                                                                     08104
(qspcreset) PROCEDURE (topflag);                                     08105
  % reset default queryspecs values %                               08106
  % qspecs is global %                                              08107
    IF topflag THEN                                                  08108
      BEGIN                                                          08109
      qspecs.quercol _ FALSE;                                        08110
      qspecs.querinc _ TRUE;                                         08111
      qspecs.queropts _ FALSE;                                       08112
      END;                                                           08113
    qspecs.querprt _ FALSE;                                          08114
    RETURN;                                                          08115
    END.
                                                                     08116
(qstrip) PROCEDURE (stid, destsw, topflag, string);                  08117
  % decrud a data base stmt %                                       08118
  % THIS IS ALMOST IDENTICAL TO QSTMT!!! DROP ONE. %                08119
  % Given the source sequence work area address, and the destination
  sequence work area address, drop off special query commands  and
  port-send the string out %                                         08120
    LOCAL                                                            08121
      code,   % Type of statement: menu, non-menu, etc. %            08122
      truncate,  % Do not execute links beyond the frist line if
```

```
      TRUE. %                                                    08123
      dirflg,    % TRUE if directive found %                    08124
      nextchr;   % curent next char to process %                08125
   LOCAL TEXT POINTER                                            08126
      oldpos,    % front of text to SEND %                      08127
      endnew;      % end of text to SEND %                      08128
   REF destsw, string;                                           08129
   % initialize %                                               08130
      oldpos _ endnew _  stid;                                   08131
      oldpos[1] _ endnew[1] _                                    08132
         CASE code _ tstmenu(stid) OF                            08133
            = 3, = 4: 2; % Comment character to be peeled off %
                                                                 08134
            ENDCASE 1; % Logical front of this statement %       08135
      truncate _ NOT topflag AND code IN [1,2];                  08136
   % Handle contents of this statement.  Optimize case where no
   directives occur.  No FIND statements used until queryspecs
   encountered. %                                                08138
      LOOP                                                       08139
         BEGIN % keep processing through this stmt.  If this is a
         top level node, process  until segments of text are sent.
         If this is a menu item, gather data in global string for
         single send when includes have been taken; avoid CRs
         generated by sends. %                                   08140
         dirflg _ qdirparse ($oldpos, $endnew : nextchr);        08141
         % append and send previous segment if this is top level
         node; otherwise, just append %                          08142
            dirflg _ qappender(&string, $"", $oldpos, $endnew,
            truncate, dirflg, destsw.swvspec.vstrnc);            08143
               % If this is a menu node, only append first 72
               characters or one line; change dirflg to reflect
               limit reached to exit loop. %                     08144
            IF topflag THEN % send out now. %                    08145
               qsender(&string, &destsw, topflag);               08146
            IF NOT dirflg THEN EXIT LOOP;                        08147
         % advance pointers for include checks and next scan %
                                                                 08148
            oldpos[1] _ endnew[1] _ nextchr;                     08149
         END; % of text/include loop %                           08150
   RETURN;                                                       08151
   END.
                                                                 08152
(qmenu) PROCEDURE (stid, endstd, destsw);                        08153
   % output a query menu %                                      08154
   % Given the stid of the up of the menu and the end stid of the
   sequence, the main display area address, and a destination work
   area address, qmenu generates the output for the plex below (if
   any).                                                         08155
   Menu statements are defined as:                               08156
      All named statements                                       08157
      All statements beginning with an asterisk                  08158
   NOTE:  All other statements will appear exactly where they occur,
   interspersed with menu selections, and at same level.         08159
   These are formatted so they become selectable by number.  Qmenu
   attaches a number to each and pushes an address to that place (an
   stid) onto the query state stack so that the mapping from option
```

```
   selection to address in a file is easy.                          08160
   Qspecs determine whether COLUMNATION is to be performed.         08161
   System globals:  qmenumax - max # simultaneous menu items
   permitted.  qcolwidth = # horizontal increments to a menucolumn. %
                                                                    08162
      LOCAL                                                         08163
         menusize,   % size of the menu before checking with user %
                                                                    08164
         menumax,    % local temp for max allowed items %          08165
         code,       % result of testing for menu statement %      08166
         stringadr,  % temp for calls %                           08167
         % **********                                             08951
            menusw,     %% sequence work area for MENU PLEX only %%
                                                                    08168
            ********** %                                          08952
         oldview[2], % temp for saving/restoring viewspecs %      08169
         origlevel;  % level of the UP of this plex (for comparison
         to detect end) %                                         08170
      LOCAL STRING                                                08171
         columnatestr [75],   % one line %                        08172
         selection[20];       % selectable number accompanying menu
         item %                                                   08173
      REF menusw, destsw;                                         08174
      % initialization %                                         08175
         menumax _ menusize _                                     08176
            IF qspecs.queropts THEN qspecs.queropts ELSE qmenumax;
                                                                    08794
         *columnatestr* _ NULL;                                   08177
      % open a new sequence for this menu plex - get first stid %
                                                                    08178
         &menusw _                                                08179
            openseq(stid, endstd, destsw.swvspec, destsw.swvsp2, 0,
            % no user sequence generator, please! % destsw.swcacode);
                                                                    08897
               % open a secondary sequence so we can control reading
               of menu plex right here %                          08896
         % **********                                             08935
            ON SIGNAL ELSE closeseq(&menusw);                       08180
            ********** %                                          08936
            ON SIGNAL ELSE closeseq(&menusw := 0);                  08937
         oldview _ destsw.swvspec;                                08181
         oldview[1] _ destsw.swvsp2;                              08182
         stid _ seqgen(&menusw);                                  08183
            % throw away top statement - already processed %      08184
         destsw.swvspec.vstrnc _ 1;                               08185
            % 1 line (truncated for simple menu case %           08186
         origlevel _ menusw.swvspec.vslev _ menusw.swclvl + 1;    08187
            % 1 level (truncated for simple menu case % % change to
            permit menued lower levels if desired. %            08188
      % process the entire plex %                                08189
         LOOP    % once for each statement %                     08191
            BEGIN                                                08192
            stid _ seqgen(&menusw);    % next stid %             08193
            % test for done (or no data) %                       08194
               IF stid = endfil THEN EXIT LOOP;                  08195
               CASE menusw.swclvl OF                             08196
```

```
            < origlevel: EXIT LOOP;    % done with menu plex %
                                                              08197
            > origlevel: REPEAT LOOP; % sub of plex level %
                                                              08198
            ENDCASE NULL;                % fall through %    08199
         % see if it's a menu statement %                     08200
           CASE code _ tstmenu(stid) OF                       08201
            = 1, = 2: % menu statements %                     08202
               BEGIN                                          08203
               % test for too many menu items - exceeds system
               max %                                          08204
                  IF (qmenucnt _ qmenucnt + 1) > menumax AND
                  nlmode=fulldisplay THEN                     08205
                     BEGIN                                    08206
                        % Go back to interact with user; must
                        come back here to close menusw or
                        continue. %                           08207
                     destsw.swstid _ endfil;                  08208
                     sport(&destsw);                          08209
                     menumax _                                08210
                        % **********                          08953
                           mencont(menumax, menusize, &menusw,
                           &destsw, $oldview);                08954
                           ********** %                       08955
                           mencont(menumax, menusize, &destsw,
                           $oldview);                         08956
                     END;                                     08211
               % Put the item on the menu stack. %           08212
                  IF newstk THEN                              08213
                     BEGIN                                    08214
                     IF NOT pushent(stid, mentyp) THEN        08215
                        BEGIN                                 08216
                           % Cannot SIGNAL or use err because
                           of switched stacks.  Rather, set
                           global error flag to be TRUE, and
                           send back endfil. %                08217
                        hseger _ TRUE;                        08218
                        % **********                          08938
                           closeseq(&menusw);                 08940
                           ********** %                       08939
                           closeseq(&menusw := 0);            08220
                        destsw.swcstid _ destsw.swstid _
                        endfil;                               08221
                           % send back EOF => we are done %
                                                              08219
                        sport(&destsw);                       08222
                           % don't want to be called again.
                           Finished with a query branch %     08919
                        END;                                  08223
                     END;                                     08224
               % build selection number string %            08225
                  % **********                               08909
                     CASE code OF                             08795
                        = 1: *selection* _ STRING(qmenucnt),
                        '., SP;                               08796
                        = 2: *selection* _ STRING(qmenucnt),
```

```
                        '.;                                          08797
                        ENDCASE;                                     08798
                      ********** %                                   08910
                      IF FIND SF(stid) SP THEN                       08911
                         *selection* _ STRING(qmenucnt), '.
                                                                     08914
                      ELSE                                           08912
                         *selection* _ STRING(qmenucnt), '., SP;
                                                                     08913
                IF qspecs.quercol THEN % columnation requested %
                                                                     08227
                   BEGIN                                             08228
                   IF code = 2 THEN                                  08229
                      BEGIN                                          08230
                         % This is an unnamed, menued statement
                         in the middle of a columnated menu.
                         Flush out any current columnation stuff
                         and continue with the first line of
                         this menued, unnamed statement. %           08231
                      IF columnatestr.L THEN                         08233
                         qcolumnate(&destsw, stid,
                         $columnatestr, $selection, TRUE);           08234
                      END    % Continue processing. %               08236
                   ELSE                                              08237
                      BEGIN                                          08238
                      qcolumnate(&destsw, stid, $columnatestr,
                      $selection, FALSE); % add this %               08239
                      EXIT CASE;                                     08240
                      END;                                           08241
                   END;                                              08242
                % straight menu %                                    08243
                   qstmt(&menusw, &destsw, FALSE, $selection);
                                                                     08244
                      % do statement; execute INCLUDES; no sub-
                      structure; append "selection" to front %
                                                                     08245
                IF overscreen THEN                                   08246
                   BEGIN                                             08247
                   menumax _                                         08248
                      % **********                                   08957
                         mencont(menumax, menusize, &menusw,
                         &destsw, $oldview);                         08959
                         ********** %                               08958
                         mencont(menumax, menusize, &destsw,
                         $oldview);                                  08915
                   overscreen _ FALSE;    % reset overscreen %
                                                                     08250
                   END;                                              08251
                END; % of menu-handling %                            08252
             = 3, = 5: % non-menu statements - may have
             directives %                                            08253
                BEGIN                                                08254
                   % flush out any current columnation stuff %
                                                                     08255
                   % process it, exclude substructure %             08256
                stringadr _                                          08257
```

```
                        IF code = 3 THEN $""                              08920
                        ELSE $selection;    % asterisk case %              08921
                     destsw.swvspec.vstrnc _ oldview.vstrnc;              08259
                        % truncation may be on - make default number
                        of lines show for this case %                     08258
                     gstmt(&menusw, &destsw, FALSE, stringadr);           08260
                     IF overscreen THEN                                   08261
                        BEGIN                                             08262
                        menumax _                                         08263
                           % **********                                   08960
                              mencont(menumax, menusize, &menusw,
                              &destsw, $oldview);                         08962
                              ********** %                               08961
                              mencont(menumax, menusize, &destsw,
                              $oldview);                                  08916
                        overscreen _ FALSE;                               08264
                        END;                                              08265
                     destsw.swvspec.vstrnc _ 1;                           08267
                        % restore the temporary suppression of line
                        truncation disabled for non-named menu or
                        non-menu %                                        08266
                     END;                                                 08268
                  = 0: NULL;    % watch out - error occurred %            08269
                  ENDCASE NULL;                                           08270
            END;    % of menu plex loop %                                08271
      % cleanup %                                                        08272
         IF columnatestr.L THEN                                           08274
            gcolumnate(&destsw, stid, $columnatestr, $selection,
            TRUE);                                                        08275
               % flush out possible incomplete line %                    08273
         % **********                                                    08941
            closeseq(&menusw);                                            08945
            ********** %                                                 08942
            closeseq(&menusw := 0);                                       08943
               % done with menu plex sequence.  Return to caller %
                                                                          08944
         destsw.swvspec _ oldview;                                        08277
         destsw.swvsp2  _ oldview[1];                                     08278
      RETURN;    % normal exit %                                         08279
      END.
                                                                          08280
(mencont) PROCEDURE (menumax, menusize, destsw, oldview);                 08281
   % ? %                                                                 08282
      % **********                                                       08963
         PROCEDURE (menumax, menusize, menusw, destsw, oldview);
                                                                          08964
         **********                                                       08965
         PROCEDURE (menumax, menusize, destsw, oldview);                  08966
         ********** %                                                    08967
      REF menusw, destsw, oldview;                                        08283
      IF moremenu THEN                                                    08284
         menumax _ menumax + menusize    % on with the next chunk %
                                                                          08286
      ELSE                                                                08288
         BEGIN                                                            08289
         % **********                                                    08947
```

```
            closeseq(&menusw);                                          08950
            ********** %                                               08948
            closeseq(&menusw := 0);                                     08949
         destsw.swvspec _ oldview;                                      08291
         destsw.swvsp2 _ oldview[1];                                    08292
         destsw.swstid _ destsw.swcstid _ endfil;                       08293
         sport(&destsw);                                                08294
         END;                                                           08295
      RETURN(menumax);                                                  08296
      END.

                                                                        08297
(tstmenu) PROCEDURE (stid);                                             08298
   % test if query menu stmt %                                          08299
   % RETURNs                                                           08300
   0: if ERROR encountered                                              08301
   1: if statement represented by stid is named (therefore menu)
                                                                        08302
   2: if statement is an unnamed menued statement                       08303
   3: if statement represented by stid starts with non-menu symbol
                                                                        08304
   4: if statement begins with comment character                        08305
   5: No "non-menu" character defined.  This is non-menu, but no
   character peeled off. %                                              08306
      LOCAL char;                                                       08307
      ON SIGNAL ELSE RETURN(0);                                         08308
      % WATCH OUT - GETNMF CALLS ERR with "Bad statement identifier"
      %                                                                 08309
      % search for special comment or menu characters %                 08310
         IF NOT menchr OR NOT comchr THEN                               08311
            BEGIN                                                       08312
            IF getnmf(stid) THEN RETURN(1)                              08313
            ELSE RETURN(5);                                             08314
            END;                                                        08315
         CCPOS SF(stid);                                                08316
         CASE (char _ READC) OF                                         08317
            =menchr: RETURN(3);                                         08318
            =comchr: RETURN(4);                                         08319
            ENDCASE                                                     08320
               IF getnmf(stid) THEN RETURN(1)                           08321
               ELSE RETURN(2);                                          08322
      END.

                                                                        08323
(qcolumnate) PROCEDURE (sw, stid, columnatestr, selection, flushflg);
                                                                        08324
   % query columnation %                                                08325
   % Given the address of a sequence work area, the "current" line
   string, the selection string for the next item, and a flag for
   just flushing out the previous information, qcolumnate either does
   the simple flush, or adds the new item (sw.stid) to the columnated
   menu.
   Obeys global qcolwidth. %                                            08326
      REF sw, selection, columnatestr;                                  08327
      LOCAL STRING names[30];                                           08328
      LOCAL savlvl, qcolwork[7]; % temp READC work area %               08329
      % see if time to pump out current line %                         08330
         IF columnatestr.L >= 72 OR flushflg THEN                       08331
```

```
            BEGIN                                                   08332
            savlvl _ sw.swclvl := sw.swslvl + 1;                    08333
            send(&sw, $columnatestr);                               08334
            sw.swclvl _ savlvl;                                     08335
            *columnatestr* _ NULL;                                  08336
            IF flushflg THEN RETURN;                                08337
            END;                                                    08338
         % construct new entry %                                    08339
           % set up for and do name extraction %                    08340
            *names* _ NULL;                                         08341
            qcolwork _ stid;                                        08342
            qcolwork[1] _ 1; % point to first char %                08343
            fechc1(forward, $qcolwork);                             08344
            xtrnam($names, $qcolwork, -1, 0);                       08345
           % add on to current line being built %                   08346
            *columnatestr* _ *columnatestr*, *selection*, *names*,
            *spacestr*[1 TO (qcolwidth - names.L - selection.L)];
                                                                    08347
         RETURN;                                                    08348
         END.
                                                                    08349
         % NOTE:  sequence work area is destination -- PRIMARY one %
                                                                    08350
(qlnkspec) PROCEDURE (tptadr, usrstg, fnmstg, stnstg, vspstg);      08351
   % Makes use of the new lnkprs %                                 08352
      LOCAL datastr[30];                                            08353
      REF  usrstg, fnmstg, stnstg, vspstg, tptadr;                  08354
      lnkprs(&tptadr, $datastr);                                    08355
      % Set up text pointers from datastr. %                       08356
         p1 _ datastr[us]; p1[1] _ datastr[us + 1];                 08357
         p2 _ datastr[ue]; p2[1] _ datastr[ue + 1];                 08358
         p3 _ datastr[fs]; p3[1] _ datastr[fs + 1];                 08359
         p4 _ datastr[fe]; p4[1] _ datastr[fe + 1];                 08360
         p5 _ datastr[ds]; p5[1] _ datastr[ds + 1];                 08361
         p6 _ datastr[de]; p6[1] _ datastr[de + 1];                 08362
         p7 _ datastr[vb]; p7[1] _ datastr[vb + 1];                 08363
         p8 _ datastr[ve]; p8[1] _ datastr[ve + 1];                 08364
         tptadr _ datastr[le]; tptadr[1] _ datastr[le + 1];         08365
      *usrstg* _ p1 p2; % user %                                    08366
         astruc(&usrstg);                                           08367
      *fnmstg* _ p3 p4; % file name %                               08368
         astruc(&fnmstg);                                           08369
      *stnstg* _ p5 p6; % statement name, number, or marker %       08370
         IF stnstg.L > empty AND *stnstg*[1] NOT= '# THEN
         astruc(&stnstg);                                           08371
      *vspstg* _ p7 p8; % view %                                    08372
      IF usrstg.L # empty AND fnmstg.L # empty THEN                 08373
         *fnmstg* _ '<, *usrstg*, '>, *fnmstg*                      08374
      ELSE IF NOT tptadr.stastr AND fnmstg.L # empty AND <NLS,
      IOEXEC, gdftdir>( tptadr.stfile, &usrstg) THEN                08817
         *fnmstg* _ '<, *usrstg*, '>, *fnmstg*;                     08818
      IF stnstg.L = empty AND fnmstg.L # empty  THEN *stnstg* _ '0;
                                                                    08380
         % May really want it to be empty, e.g., to change viewspecs
         when not changing file %                                   08381
      RETURN;                                                       08382
```

```
      END.
                                                                    08383
(ofstid) PROCEDURE (dpa, stnstg);                                   08384
   % evaluate  intra-file address expression %                      08385
   % Given the display area address in order to get the file number
   and a string containing an address expression, this routine will
   open the file, and return the corresponding stid and character
   count. %                                                         08386
      LOCAL vs1, vs2, cacode, useqgen;                              08387
      LOCAL TEXT POINTER stptr, z1, z2;                             08388
      REF dpa, stnstg;                                              08389
      stptr _ origin;                                               08390
      stptr.stfile _ dpa.dacsp.stfile;                              08391
      FIND SF(*stnstg*) ^z1 SE(*stnstg*) ^z2;                       08392
      vs1 _ caddexp($z1, $z2, &dpa, $stptr : vs2, cacode, useqgen);
                                                                    08393
      RETURN(stptr, stptr[1], vs1, vs2, cacode, useqgen);           08394
      END.
                                                                    08395
(feedsw) PROCEDURE (sw, astrng);                                    08396
   % Given the address of a sequence work area, this routine changes
   its vspecs in accord with the specifications in the A-string
   passed it.  It passes the characters in the A-string to <PRMSPC,
   SETLT>, except for content analyzer patterns. %                  08397
      LOCAL count, length, char, vs1, vs2;                          08398
      LOCAL TEXT POINTER tp;                                        08399
      REF sw, astrng;                                               08400
      length _ astrng.L;                                            08401
      count _ empty - 1;                                            08402
      vs1 _ sw.swvspec;                                             08403
      vs2 _ sw.swvsp2;                                              08404
      UNTIL (count _ count+1) > length DO                           08405
         IF (char _ *astrng*[count]) = '; THEN                      08406
            BEGIN                                                   08407
            UNTIL (char _ *astrng*[count _ count + 1]) = '; DO      08408
               BEGIN                                                08409
               % skip over ca pattern. %                            08410
               IF count >= length THEN EXIT LOOP1;                  08411
               END;                                                 08412
            END                                                     08413
         ELSE vs1 _ setlt(char, vs1, vs2 : vs2);                    08414
      sw.swvspec _ vs1;                                             08415
      sw.swvsp2 _ vs2;                                              08416
      RETURN;                                                       08417
      END.
                                                                    08418
                                                                    08419
% Context stack building %                                         08420
   (pushent) PROCEDURE (stid, type);                                08421
      % Makes use of the global curstk as the context to be filled in %
                                                                    08422
      % Type = 1: mentyp; =2: multyp. %                            08423
      LOCAL stkad, itemad, wrkcnt, work;                            08424
      REF stkad, itemad;                                            08425
      IF NOT curstk.stktyp THEN                                     08426
         % get a new stack %                                        08427
```

```
      BEGIN                                                          08428
      % get first block in this stack; link it to conrng %           08429
      IF NOT (&stkad _ getblk(67B, &qstorblk)) THEN                  08430
         BEGIN %we are out of free storage - see if more is
         available%                                                  09001
         IF NOT (&qstorblk _ getblk(1777B , $dspblk)) THEN
         RETURN(FALSE);                                              09002
         &qstorblk _ &qstorblk + bhl;                                09003
         makezone(&qstorblk, 70B, 70B, 1777B);                       09004
         IF NOT (&stkad _ getblk(67B, &qstorblk)) THEN               09005
            RETURN(FALSE);                                           09006
         END;                                                        08431
      stkad[1] _ &qstorblk;                                          08432
      curstk.constk _ &stkad _ &stkad + 2;                           08433
      stkad.conlnk _ -1;                                             08434
      stkad.stkcnt _ 0;                                              08435
      curstk.stktyp _ type;                                          08436
      END                                                            08437
   ELSE                                                              08438
      BEGIN                                                          08439
      % Use totcnt to get to the proper block. %                     08440
      &stkad _ curstk.constk;                                        08441
      FOR wrkcnt _ curstk.totcnt DOWN stkmax UNTIL <= stkmax DO
                                                                     08442
         BEGIN                                                       08443
         IF (&stkad  _ stkad.conlnk)<=0 THEN RETURN(FALSE);          08444
         END;                                                        08445
      END;                                                           08446
   % Put the entry on the stack. %                                   08447
      IF (stkad.stkcnt _ stkad.stkcnt+1) > stkmax THEN               08448
         BEGIN                                                       08449
         % must get a new block %                                    08450
         % Reset stkad.stkcnt %                                      08451
            BUMP DOWN stkad.stkcnt;                                  08452
         % Try to get a new block.  Put address in previous
         block's link field. %                                       08453
         IF NOT (work _ getblk(67B, &qstorblk)) THEN                 09007
            BEGIN %we are out of free storage - see if more is
            available%                                               09008
            IF NOT (&qstorblk _ getblk(1777B , $dspblk)) THEN
            RETURN(FALSE);                                           09009
            &qstorblk _ &qstorblk + bhl;                             09010
            makezone(&qstorblk, 70B, 70B, 1777B);                    09011
            IF NOT (work _ getblk(67B, &qstorblk)) THEN              09012
               RETURN(FALSE);                                        09013
            END;                                                     09014
         stkad.conlnk _ work + 2;                                    08456
         &stkad       _ work;                                        08457
         stkad[1]     _ &qstorblk;                                   08458
         &stkad       _ &stkad + 2;                                  08459
         stkad.conlnk _ -1;                                          08460
         stkad.stkcnt _ 1;                                           08461
         END;                                                        08462
      &itemad _ &stkad + stkad.stkcnt*2;                             08463
      itemad _ stid;                                                 08464
      itemad[1] _ IF getnmf(stid) THEN getnam(stid)                  08465
```

```
            ELSE 0; % zero if no name %                                  08466
          BUMP curstk.totcnt;                                            08467
        RETURN(TRUE);                                                    08468
        END.
                                                                         08469
FINISH of chelp                                                          08471
```

```
< NLS, CMLCOMP.NLS;11, >, 11-DEC-74 09:54 CHI ;;;; [[ add NOT variable]]
FILE cml CHECK % meta <nsw-sources>cml.rel %% (meta,)
(nsw-sources,cml.rel,) %                                                02
   META file                                                            03
      % *** Chage date wen compiling in (file) *** %                   0388
   % DECLARATIONS %                                                     04
      DUMMY: alter  succ option anyof dumy;                             05
      ERROR: -> "END." $(subsys) "FINIS" :fin[]*;                       06
      SIZE: S=3000 M=100 K=100 N=1000 L=300 G=100;                      07
      SET:                                                              08
         % RECOGNIZERS %                                               09
           KEYOP=1B10  % keyword recognition operator %                 010
           CONFIRM=2B10        % get command confirmation %             011
           SSEL=3B10   % source selection %                            012
           DSEL=4B10   % destination selection %                       013
           LSEL=5B10   % literal selection %                           014
           CA=6B10             % CA char %                             015
           VIEWSPECS=7B10      % gets viewspecs %                      016
           LEVADJ=10B10        % get level adjust string %             017
         % OPTIONAL ELEMENTS %                                         018
           OPTION=11B10        % optional parameter %                  019
           ANYOF=12B10 % repeated list with failure %                  020
         % CONTROL ELEMENTS %                                          021
           PFCALL=21B10        % parsing function %                    022
           EXECUTE=22B10       % transfer to another point in tree %
                                                                       023
           CALL=23B10  % subroutine call %                             024
         % FEEDBACK ELEMENTS %                                        025
           FBCLEAR=31B10       % clear feedback buffer %               026
           ECHO=32B10  % echo noise word string %                      027
           RECHO=33B10 % replace last thing echoed %                   028
         % VALUE MANIPULATIONS %                                       029
           STORE=41B10 % store value into variable %                   030
           LOAD=42B10          % load variable to ptr stack %          031
           ENTER=43B10 % enter constant into stack %                   032
           VALUEOF=44B10;      % valueof built in function %           033
      FLAGS:                                                            034
         attr             % attribute values %                         035
         initloc          % ptr to INITIALIZATION statement %          036
         termloc          % ptr to TERMINATION statement %             037
         rentryloc        % ptr to RENTRY statement %                  038
         sav              % id value for command %                     039
         value            % temporary accumulator %                   040
         ctlbits                % temporary accumulator %              041
         top              % address of last generated node %          042
         alt              % address of alternative node %             043
         suc;             % address of successor node %                044
      FIELDS: OP=[6:30] CTL=[3:27]                                      045
         VAL=[9:18] LH=[18:18] RH=[18:0] ;                              046
      ATTRIBUTES:                                                       047
         pfunctname             % name of external parsing function %
                                                                       048
         functname              % name of external L10 function %     049
         rulename               % name of a parse rule %              050
         varname                % temporary variable %                051
         litstr;                % .ID has literal associated with it %
```

```
                                                                    052
% PARSING RULES %                                                   053
   % file definition %                                              054
      file = "FILE" .ID <"-CML COMPILER 12/6/74">                   055
         <"-FILE "*1> &NAME &DISCARD                                056
         :[1,] % output a dummy word %                              057
         $dcls $rule   % 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 L10 string for all keyword strings which
               are not external %                                   064
                  $SYMS(                                            065
                     ?NOT ?@ defned *$ (                            066
                        ?@ litstr *$                                067
                           (?NOT ?@ extern *$                       068
                              >*$                                   069
                              *L$^LH *L$,                           070
                              *S$                                   071
                           /TRUE)                                   072
                        /TRUE)                                      073
                     /TRUE)                                         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
         ( "COMMAND" "WORD"  #<',> (.SR '= .NUM :dclcw[2]*) ';      088
         / [dclattr]  .#<',> dclitem :dcllist[$] '; *);             087
      dclcw [-,-] => >*1 _ LSH(*N2)18+. @S reloca *1 *L1^LH *L1, *S1;
                                                                    089
      dclattr = % declaration attributes %                          090
         ("VARIABLE" &attr _ 1                                      091
         / "FUNCTION" &attr _ 2                                     092
         / "PARSEFUNCTION" &attr _ 3                                093
         / "EXT-KEYWORD" &attr _ 4                                  094
         / "EXTERNAL" &attr _ 5);                                   095
      dclitem = .ID/ .SR [?IF attr # 4 &FAIL];                      096
      dcllist [$] =>                                                097
         ( ?IF attr = 1 % VARIABLE %                                098
            $(@S varname *$ >*$ &BSS 1, )                           099
         / ?IF attr = 2 % FUNCTION %                               0100
            $(@S functname *$ )                                    0101
         / ?IF attr = 3 % PARSEFUNCTION %                          0102
            $(@S pfunctname *$ )                                   0103
         / ?IF attr = 4 % EXT-KEYWORD %                            0104
```

```
          $(@S extern *$ @S litstr *$ )                                    0105
        / ?IF attr = 5 % EXTERNAL %                                        0106
          $(@S extern *$ ));                                               0107
   % SUBSYSTEM definition %                                                0108
     subsys = "SUBSYSTEM" .ID &LABEL                                       0109
        "KEYWORD" .SR :subs2[ 2 ]                                          0110
        &MARK &sav _ 0 &initloc _ 0  &termloc _ 0  &rentryloc _ 0
                                                                           0111
        #(command / rule  )                                                0112
        &UNMARK &top _ sav *  "END.";                                      0113
     subs2 [.ID, .SR] =>                                                   0114
        % define the subsystem identifier as an external symbol
        whose value is the address of the subsystem dispatch record
        %                                                                  0115
          >^*1                                                             0116
        % output the subsystem dispatch record %                          0117
          % word 1: RH = ptr to subsystem name string %                    0118
            =*S2,                                                          0119
          % word 2: RH = ptr to start of COMMANDS %                       0120
          % word2: LH = validation code %                                 0121
            110473^LH top^RH\1                                             0122
          % word 3: LH = ptr to termination rule/0, RH = ptr to
          initialization rule /0 %                                         0123
            header[ =termloc, =initloc ]                                   0124
          % word 4: LH = ptr to rentry rule/0 %                           0125
            header[ =rentryloc, =0 ];                                      0126
   % command definitions %                                                0127
     % 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 %                    0128
     command =                                                             0129
         ( "COMMAND"                                                       0130
           .ID &LABEL :defcmd[1] &MARK                                     0131
           '= exp &alt_sav  &suc_0  :eval[1] * &sav _ top                  0132
           &UNMARK *  ';                                                   0133
         / "INITIALIZATION"                                                0134
           :[?IF initloc # 0 &FAIL]                                        0135
           rule &initloc _ top                                             0136
         / "TERMINATION"                                                   0137
           :[?IF termloc # 0 &FAIL]                                        0138
           rule &termloc _ top                                             0139
         / "RENTRY"                                                        0140
           :[?IF rentryloc # 0 &FAIL]                                      0141
           rule &rentryloc _ top ) ;                                       0142
     rule =                                                                0143
        .ID &LABEL                                                         0144
        '= exp &alt_0  &suc_0  :eval[1] *                                  0145
         :defcmd[1] *  '; ;                                                0146
     defcmd                                                                0147
        [.ID]       =>                                                     0148
           [  ( ?@ varname *1                                              0149
              / ?@ functname *1                                            0150
              / ?@ pfunctname *1 )                                         0151
              <"previously defined name used as a rule" *1 LOC LINE>
              &FAIL]                                                       0152
           rulname[*1];                                                    0153
```

```
% expression definition %                                              0154
   exp = .#<'/>subexp :alter[$];                                       0155
   subexp =                                                            0156
      .#factor :succ[$];                                               0157
   factor =                                                            0158
      '( exp ')/                                                       0159
      '[ exp '] :option[1]/                                            0160
      "DUMMY"                                                          0161
         [ +'/ <"DUMMY not last alternative"> &FAIL]                   0162
         :dumy[]/                                                      0163
       term ;                                                          0164
% output production rules %                                           0165
   % 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. %                                                     0166
   % 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. %                                                            0167
   %--------%                                                          0168
   eval                                                                0169
      % process a sequence of alternatives %                           0170
       [alter] =>                                                      0171
          ?*Q1 = 1 % single subnode %                                  0172
             eval[*1:1]                                               0173
          / % multiple subnodes %                                      0174
             $*1_( % select all alternatives beginning with the
             last, invoking the eval rule recursively to process
             them.%                                                   0175
                ! suc ( eval[*$] )                                    0176
                &alt _ top                                            0177
                );                                                    0178
      % process a sequence of successors %                            0179
       [succ] =>                                                      0180
          ?*Q1 = 1 % single subnode %                                  0181
             eval[*1:1]                                               0182
          / % multiple subnodes %                                      0183
             $*1_( % select all successors beginning with the
             last, invoking the eval rule recursively to process
             them.%                                                   0184
                ((?LAST  eval[*$])                                    0185
                / (! alt (&alt _ 0 eval[*$]) ))                       0186
                &suc _ top                                            0187
                );                                                    0188
      % generate code for an optional expression %                    0189
       [option[-]]      =>                                            0190
          ! alt, suc ( &alt _ 0 &suc _ 0   eval[*1:1] )               0191
          header[=suc, =suc]                                          0192
          OPTION top^RH\1                                             0193
          &top _ .-2;                                                 0194
      % generate code for a '$ expression %                           0195
       [anyof[-]]       =>                                            0196
          ! alt, suc ( &alt _ 0 &suc _ 0   eval[*1:1] )               0197
          header[=suc, =suc]                                          0198
```

```
            ANYOF top^RH\1                                          0199
            &top _ .-2;                                             0200
         % generate code for a function %                           0201
          [function] => *1;                                         0202
         % generate code for an assignment %                        0203
          [assign] => *1;                                           0204
         % generate code for a loop construct %                     0205
          [perform] => *1;                                          0206
         % set alternative  for DUMMY node %                        0207
          [dumy] => &top _ suc;                                     0208
         % generate code for any terminal nodes. %                  0209
          [-] =>                                                    0210
            &top _ .                                                0211
            header[=suc, =alt]                                      0212
            *1;                                                     0213
%---------                                                         0214
(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 %                 0215
   header                                                           0216
      [0,0]       => 0\0;                                         0217
      [0,-]       => 0^LH  *N2^RH\1;                              0218
      [-,0]       => *N1^LH  0^RH\2;                              0219
      [-,-]       => *N1^LH  *N2^RH\3;                            0220
% terminal type  productions %                                     0221
   assign                                                           0222
      [.ID,-] =>                                                    0223
         !alt (&alt _ 0 eval[store[*1]])                            0224
         &suc _ top                                                 0225
         eval[ *2 ]                                                 0226
         &suc _ top;                                                0227
   call                                                             0228
      [.ID] =>                                                      0229
         CALL *V1,;                                                 0230
      [.ID,-] =>                                                    0231
         CALL *N2^VAL *V1,;                                         0232
   echo[.SR] =>                                                     0233
      @S noddt *1                                                   0234
      ECHO =*S1,;                                                   0235
   endcmd[] =>                                                      0236
      CONFIRM,;                                                     0237
   enter                                                            0238
      [-] =>                                                        0239
         ENTER *N1,;                                                0240
   execute[.ID] =>                                                  0241
      EXECUTE *V1,;                                                 0242
   fbclear[] =>                                                     0243
      FBCLEAR,;                                                     0244
   function                                                         0245
      [.ID,succ] =>                                                 0246
         % check for id used as a rule name or as a variable name
         %                                                          0247
            [ (?@ varname *1/ ?@ rulename *1) <"ERROR: illegal use
            of rule/variable name: " *1 LOC LINE> ]                 0248
         % mark the .ID with the "functname" attribute if not set
         to pfunctname, generate the appropriate type of call  %
```

```
                                                                    0249
            (?@ pfunctname *1 eval[pfcall[*1,=*Q2]]                 0250
            / @S functname *1 eval[call[*1,=*Q2]])                  0251
         &suc _ top                                                 0252
         !alt ( &alt _ 0 eval[*2])                                  0253
         &suc _ top;                                                0254
      [internal,succ] =>                                            0255
         eval[*1]                                                   0256
         &suc _ top                                                 0257
         !alt ( &alt _ 0 eval[*2])                                  0258
         &suc _ top;                                                0259
   internal[-] =>                                                   0260
      *N1,;                                                         0261
   keyid[.ID] =>                                                    0262
      KEYOP =*S1,;                                                  0263
   keyw[.SR,-,-] =>                                                 0264
      @S litstr *1  % mark it as a literal %                        0265
      KEYOP *N2^CTL *N3^VAL (?@defned *1 *R1\1 / *V1,);             0266
   perform % looping construct %                                    0267
      [.ID,-,-] =>                                                  0268
         % check to make sure .ID is a rule %                       0269
            [(?NOT ?@ defned *1 @S  rulename *1                     0270
            /?NOT ?@ rulename *1 <"ERROR: ID must be a rule name"
            *1 LOC LINE> &FAIL)]                                    0271
         % generate code to (1) EXECUTE the rule and (2) evaluate
         the test express*on.  Like all CML code, this sequence is
         generated in reverse order so that tree meta can do the
         appropriate linking %                                      0272
            ! alt, suc ( &alt _ 0                                   0273
               eval[ succ[ *2 ] ]                                   0274
               &suc _ top &alt _ . eval[ execute[*1] ] )            0275
            &suc _ top;                                             0276
   pfcall                                                           0277
      [.ID] =>                                                      0278
         PFCALL *V1,;                                               0279
      [.ID,-] =>                                                    0280
         PFCALL *N2^VAL *V1,;                                       0281
   recho[.SR] =>                                                    0282
      @S noddt *1                                                   0283
      RECHO =*S1,;                                                  0284
   store[.ID] =>                                                    0285
       STORE *V1,;                                                  0286
   valueof                                                          0287
      [.SR] =>                                                      0288
         @S litstr *1                                               0289
         ENTER (?@defned *1 *H1^VAL *R1\1 / *V1,);                  0290
% terminal nodes for compiler %                                     0291
   term =                                                           0292
      (subname/ confirm/ feedback/ recognition/ loop) ;             0293
   subname =                                                        0294
      .ID                                                           0295
          (('_ % assignment statement %                             0296
            % check lhs variable type %                             0297
               (?@ defned                                           0298
                  ( ?@ varname                                      0299
                  /  <"ERROR: illegal lhs variable: " *1 LOC LINE>
```

```
                &FAIL)                                                  0300
              / @S varname  @S extern)                                  0301
            param  :assign[2] )                                         0302
          / ('( .$<',>param :succ[$] ')                                 0303
            :function[2] ) % function invocation %                      0304
          / :loadid[1]  ) ;                                             0305
          loadid                                                        0306
            [.ID] =>                                                    0307
              (?@ rulename *1                                           0308
                execute[*1])                                            0309
              / (?@ varname *1                                          0310
                load[*1])                                               0311
              / (?NOT ?@ defned *1                                      0312
                @S rulename *1                                          0313
                execute[*1])                                            0314
              / (?@ litstr *1                                           0315
                @S rulename *1                                          0316
                execute[*1])                                            0317
              / <"ERROR: illegal use for identifier:  " *1 LOC
              LINE>;                                                    0318
          load                                                          0319
            [.ID] =>                                                    0320
              (?@ varname *1 LOAD *V1,);                                0321
recognition = keyword/ builtinrec;                                      0322
    keyword = .SR &ctlbits _ 7B % set TNLS & DNLS & level1 bits
    % (?@defned &value _ *H1 / &value _ 0)                              0323
      [ '! #qualifier '! ]                                              0324
      :keyw[1, =ctlbits, =value];                                       0325
    builtinrec =                                                        0326
      ( ("SSEL" &value _ SSEL/                                          0327
        "DSEL" &value _ DSEL/                                           0328
        "LSEL" &value _ LSEL)                                           0329
        '( param  :function[internal[=value],succ[1]] ') )
                                                                        0330
      /                                                                 0331
        "CA" :internal[CA]/                                             0332
        "VIEWSPECS" :internal[VIEWSPECS]/                               0333
        "LEVADJ" :internal[LEVADJ];                                     0334
feedback =                                                              0335
    '<                                                                  0336
      (  "..." .SR '> :recho[1] /                                       0337
         .SR       '> :echo[1] )  /                                     0338
    "CLEAR" :fbclear[];                                                 0339
confirm =                                                               0340
    "CONFIRM"                                                           0341
    :endcmd[] ; % call routine to terminate cmd %                       0342
loop =                                                                  0343
    "PERFORM"  .ID                                                      0344
      "UNTIL"  &value _ 1                                               0345
      '( exp ') :perform[2,=value];                                     0346
qualifier =                                                             0347
    ("L2" &ctlbits _ ctlbits-4B % clear level1 bit %/                   0348
    "NOTD" &ctlbits _ ctlbits-2B % clear DNLS bit %/                    0349
    "NOTT" &ctlbits _ ctlbits-1B % clear TNLS bit %/                    0350
    .NUM &value _ *N1 &DISCARD);                                        0351
param =                                                                 0352
```

```
            "VALUEOF" '( .SR :valueof[1] ')                                  0353
            / '# .SR :valueof[1]                                             0354
            / "NULL" :enter[=0]                                              0355
            / "TRUE" :enter[=1]                                              0356
            / "FALSE" :enter[=0]                                             0357
            / factor;                                                        0358
    END of CML                                                               0359
% runfil to create a new cml compiler %                                      0360
    DEL CML.SAV;*<ESC>                                                       0361
                                                                             0362
    EXP                                                                      0363
                                                                             0364
    TENLDR                                                                   0365
    /S                                                                       0366
    /M                                                                       0367
    /4000100                                                                 0368
    <META>LIBE                                                               0369
    CML                                                                      0370
    <ESC>                                                                    0371
    DDT                                                                      0372
    MOVE 1,116<ESC>X                                                         0373
    MOVEM 1,400000<ESC>X                                                     0374
    INITLL<ESC>G                                                             0375
    MESSAGE                                                                  0376
    JUNK                                                                     0377
                                                                             0378
                                                                             0379
    SS 400 440 CML.SAV;<ESC>                                                 0380
                                                                             0381
    DEL JUNK.REL;1<ESC>                                                      0382
                                                                             0383
    DEL XXX.REL;*<ESC>                                                       0384
                                                                             0385
    EXP                                                                      0386
                                                                             0387
```

COLSORT

```
< NLS, COLSRT.NLS;5, >, 31-MAR-77 13:40 KJM ;;;;< NLS, COLSRT.NLS;58, >,
12-JUN-74 16:29 KEV ;
FILE colsrt  % L10 to <rel-nls>colsrt %% (L10,) (rel-nls,colsrt.rel,) %
                                                                        0106
%Declarations%                                                          0107
   REF colda, tda,colsw;                                                0109
   DECLARE  less = -1, gtr = 1, sucdir = sucstr, random = 4;            0110
   DECLARE ovrflo = 0;                                                  0189
   DECLARE FIELD r1left = [1,18:18];                                    0384
   SET runtm = 15B, nout = 224B;                                        0286
%...new stuff for new sort-merge primitives...%                         0398
   REF keypr,vcvulp;                                                    0399
   %v   %                                                               0400
   %vcv   %                                                             0401
   %vcvtp  %                                                            0402
   %merge-update working size parameters%                               0403
      %mersiz  block size%                                              0404
      %merrff  merge block reformat flag%                               0405
   %update cells%                                                       0406
      %vcvu  key value for present call%                                0407
      %vcvul  list of stid's for decision proc%                         0408
      %vcvulp end+1 of vcvul%                                           0409
%...REGISTER and OP-CODE definitions...%                                0385
   REGISTER                                                             0386
      r0 = 0, r1 = 1, r2 = 2, r3 = 3, r4 = 4, r7 = 7, s = 9, m = 10, a1
      = 12, r5 = 5,                                                     0387
      a2 = 13, a3 = 14, a4 = 15;                                        0388
   SET %op codes%                                                       0389
      MOVSI = 205B, CAMN = 316B, AOBJN = 253B, EXCH = 250B,             0390
      MOVEM = 202B, IMULI = 221B, ADDI = 271B, IDIVI = 231B,            0391
      JSYS = 104B, ADD = 270B, BLT = 251B, SOSG = 377B, MOVE =          0392
      200B, MOVN = 210B, PUSH = 261B, HRL = 504B, HRRI = 541B,          0393
      HRLI = 505B, LSHC = 246B, LSH = 242B, ROT = 241B,                 0394
      JRST = 254B, JSR = 264B, AOJ = 340B,                              0395
      XCT = 256B, JFCL = 255B,                                          0396
      POPJ = 263B, POP = 262B;                                          0397
%..Default key procedure...%                                           0410
   DECLARE FIELD chif=[0,7:35];                                         0411
   (defkey)PROCEDURE(stid,outb,num);                                    0412
      LOCAL pst,val,flg,char,cnt,wdcnt,firstf;                          0413
      REF outb;                                                         0414
      CCPOS SF(stid);                                                   0415
      wdcnt _ 0;                                                        0416
      flg _ FALSE;                                                      0417
      firstf _ TRUE;                                                    0800
      WHILE num > 0 DO BEGIN                                            0418
         cnt_0;                                                         0419
         val_0;                                                         0420
         pst_chif+$val; %pointer into val%                              0421
         WHILE cnt<5 AND NOT flg DO BEGIN                               0422
            IF ((char_READC) = '@ AND NOT firstf) OR char = ENDCHR THEN
            BEGIN                                                       0423
               flg_TRUE;                                                0424
               EXIT; END;                                               0425
            ^pst_IF char IN ['a,'z] THEN char-40B ELSE char;            0426
            BUMP cnt;                                                   0427
```

```
            firstf _ FALSE;                                                 0801
            END;                                                            0428
         outb_val;                                                          0429
         BUMP wdcnt;                                                        0430
         BUMP &outb;                                                        0431
         BUMP DOWN num;                                                     0432
         IF flg THEN EXIT;                                                  0433
         END;                                                               0434
      RETURN (flg,wdcnt); END.                                              0435
%...On-line Commands...%                                                    0436
   (kprget)PROC;  %get key procedure address%                               0756
      LOCAL kpr;                                                            0757
      REF kpr;                                                              0758
      &colda _ dsparea(lcda());                                             0728
      IF (&kpr _ colda.daukeycode) = 0 THEN &kpr _ $defkey;                 0730
      RETURN (&kpr);                                                        0760
      END.                                                                  0759
   (colgdest)PROC(stid);                                                    0751
      IF getfhd(stid) THEN RETURN (getup(stid),levdown)                     0752
         ELSE RETURN (getprd(stid),levsuc);                                 0753
      END.                                                                  0754
   (sortbranch)PROC(stidx);                                                 0761
      LOCAL stid1,stid2;                                                    0762
      IF (stid1_getsub(stidx))=stidx THEN RETURN;                           0763
      stid1_plxset (stid1 : stid2);                                         0764
      sort (stidx,stid1,stid2,levdown,kprget(),FALSE);                      0765
      RETURN; END.                                                          0766
   (sortgrp)PROC(bug1,bug2);                                                0725
      LOCAL stid1,stid2,stidx,rlevcnt;                                      0726
      stid1 _ grptst (bug1, bug2 : stid2);                                  0731
      stidx_colgdest (stid1 : rlevcnt);                                     0755
      sort (stidx,stid1,stid2,rlevcnt,kprget(),FALSE);                      0740
      RETURN; END.                                                          0741
   (sortplx)PROC(bug1);                                                     0742
      LOCAL stid1,stid2;                                                    0743
      stid1 _ plxset(bug1 : stid2);                                         0748
      sort(getup(stid1),stid1,stid2,levdown,kprget(),FALSE);                0749
      RETURN; END.                                                          0750
%...Sort and friends...%                                                    0815
   (sort) PROC(stidx,stid1,stid2,rlevcnt,kpr,copyf);                        0583
      LOCAL i,cxxg,stid,stidi;                                              0584
      REF cxxg;                                                             0585
      %map out pages for buffer%                                            0839
         hmapout();                                                         0840
      ON SIGNAL ELSE hmapin();                                              0886
      vcvtp _ $vcv;                                                         0586
      stid1_grptst(stid1,stid2:stid2);                                      0588
      IF copyf                                                              0823
         THEN                                                               0824
            stid1 _ ccopgro(stidx,rlevcnt,stid1, stid2, FALSE, 0: stid2)    0825
         ELSE cmovgro(stidx,rlevcnt,stidi_stid1,stid2,FALSE,0);             0834
      &keypr_kpr;                                                           0589
      vtop_0;                                                               0590
      WHILE (stid_gsstid($stidi,stid2))#endfil DO a3ddrec(stid);            0591
      i_vtop+1;                                                             0592
```

```
   WHILE (i_i-1) >= 2 DO BEGIN                                          0593
      s3iftup (1,i);                                                    0594
      v[1]_v[i]:=v[1];                                                  0595
      END;                                                              0596
   %rearrange fast-fast-fast%                                           0821
      IF (stid_[v[vtop]+4]) # stid2 THEN                                0819
         ctragro (stid, stid, stid2, stid2, FALSE, 0);                  0865
      IF stid = stid1 THEN stid1 _ stid2;                               0835
      IF (stid_[v[1]+4]) # stid1 THEN                                   0820
         ctragro (stid, stid, stid1, stid1, FALSE, 0);                  0866
      FOR i_2 UP UNTIL > vtop DO stosuc (stid, stid _ [v[i]+4]);        0597
   %map nls pages back in%                                              0841
      hmapin();                                                         0842
   RETURN;                                                              0601
   END.                                                                 0602
(s3iftup)PROC(i,n);                                                     0603
   %sort tree into vector...from floydd%                                0604
   LOCAL j;                                                             0605
   v_v[i];                                                              0606
   LOOP                                                                 0607
      IF (j _ i+i) <= n THEN BEGIN                                      0608
         IF j<n THEN IF  c3cmpare(j, j+1) THEN BUMP j;                  0609
         IF NOT c3cmpare(j, 0) THEN BEGIN                               0610
            v[i:=j] _ v[j]                                              0611
            END ELSE EXIT;                                              0612
      END ELSE EXIT;                                                    0613
   v[i] _ v;                                                            0614
   RETURN END.                                                          0615
(a3ddrec)PROC(stid);                                                    0845
   %add stid into vector%                                               0846
   LOCAL index, index1, wdcnt;                                          0847
   IF (index _ vtop _ vtop + 1) > vmax THEN SIGNAL(sorterr,$"Too Many
   Input statements");                                                  0848
   v[vtop] _ vcvtp;                                                     0849
   [vcvtp+3] _ keypr (stid, vcvtp,3 : wdcnt);                           0850
   [vcvtp+4]_stid;                                                      0851
   WHILE wdcnt < 3 DO [vcvtp + (wdcnt:=wdcnt+1)] _ 0;                   0852
   vcvtp _ vcvtp+5;                                                     0853
   WHILE (index1 _ index / 2) > 0 DO                                    0854
      IF NOT  c3cmpare(index, index1) THEN BEGIN                        0855
         v[index1] _ v[index := index1] := v[index1]                    0856
         END ELSE EXIT;                                                 0857
   RETURN END.                                                          0858
                                                                        0859
(iodkey)PROC(n,stid);                                                   0685
   LOCAL vcvn,wdcnt;                                                    0686
   vcvn _ n*mersiz + $vcv;                                              0687
   v _ vcvn+2;                                                          0688
   IF ([vcvn] _ stid) = endfil THEN BEGIN                               0689
      [vcvn+1] _ 777776000001B;                                         0690
      [vcvn+2] _ 377777777777B; %sentinal%                              0691
      RETURN (FALSE);                                                   0692
      END;                                                              0693
   [vcvn+3]_[vcvn+4]_0;                                                 0837
   !PUSH s,r1;                                                          0694
   r1 _ keypr (stid, vcvn+2, 3 : wdcnt);                                0695
```

```
   .r1left _ -wdcnt;                                                    0696
   [vcvn+1] _ r1;                                                       0697
   !POP s,r1;                                                           0698
   RETURN (TRUE);                                                       0699
   END.                                                                 0700
(gsstid)PROC (st1,st2);                                                 0701
   LOCAL stid;                                                          0702
   REF st1;                                                             0703
   IF (stid_st1)#endfil THEN BEGIN                                      0704
      IF stid = st2 THEN st1 _ endfil ELSE st1_getsuc(stid);            0705
      END;                                                              0706
   RETURN (stid);                                                       0707
   END.                                                                 0708
(c3cmpare)PROC(i1,i2);                                                  0527
   LOCAL stid1,stid2,skcnt,corder;                                      0528
   corder_ (IF (r1 _ v[i1]) < (r2 _ v[i2]) THEN 1 ELSE 0);              0529
   !HRLI r1,-3;                                                         0836
   (ccma): !MOVE r3,0(r1); !CAMN r3,0(r2); GOTO ccme;                   0530
   a1 _ 0; !CAMG r3,0(r2); BUMP a1; RETURN;                             0531
   (ccme): BUMP r2; !AOBJN r1,ccma;                                     0532
   stid1 _ [r1+1];                                                      0533
   stid2 _ [r2+1];                                                      0534
   r2 _ r2 - 1000001B;                                                  0535
   skcnt _ 3;                                                           0536
   GOTO ccmxh;                                                          0537
   (ccmxa): BUMP skcnt;                                                 0538
      !MOVE r3,0(r1); !CAMN r3,0(r2); GOTO ccmxe;                       0539
      a1 _ 0; !CAMG r3,0(r2); BUMP a1; RETURN;                          0540
   (ccmxe): !AOBJP r1,ccmxh;                                            0541
   (ccmxf): !AOBJN r2,ccmxa;                                            0542
      IF kysrdone (stid2,r2,skcnt:r2) THEN RETURN (FALSE);              0543
      GOTO ccmxa;                                                       0544
   (ccmxh):                                                             0545
      !PUSH s,r2;                                                       0546
      IF kysrdone (stid1,r1,skcnt:r1) THEN BEGIN                        0547
         !POP s,r2;                                                     0548
         !AOBJN r2,ccmxha;                                              0549
         IF kysrdone (stid2,r2,skcnt:r2) THEN RETURN (corder);          0550
         (ccmxha): RETURN (TRUE);                                       0551
         END;                                                           0552
      !POP s,r2;                                                        0553
      GOTO ccmxf;                                                       0554
   END.                                                                 0555
(kysrdone)PROC(stid,loc,skcnt);                                         0556
   LOCAL wdcnt,done;                                                    0557
   REF loc;                                                             0558
   IF loc > 0 THEN RETURN (TRUE);                                       0559
   IF loc = 0 THEN BEGIN                                                0560
      IF vcvtp > vcvend THEN SIGNAL (sorterr, $"Sort buffer full,
      aborted");                                                        0817
      !PUSH s,r1;                                                       0561
      IF skcnt>0 THEN BEGIN                                             0562
         r1_vcvtp-skcnt;                                                0563
         .r1left_-skcnt;                                                0564
         (kysra): !PUSH s,0(r1); !AOBJN r1,kysra;                       0565
         END;                                                           0566
```

```
        done _ keypr (stid,vcvtp-skcnt,skcnt+10:wdcnt);                    0567
        IF skcnt>0 THEN BEGIN                                              0568
           r1 _ vcvtp; .r1left_ skcnt; !TRNA 0,0;                          0569
           (kysrb): !POP s,0(r1);                                          0570
           r1 _ r1 - 1000001B;                                             0571
           !JUMPGE r1,kysrb;                                               0572
           END;                                                            0573
        r1 _ vcvtp; .r1left_skcnt-wdcnt;                                   0574
        vcvtp _ vcvtp+wdcnt-skcnt;                                         0575
        loc_r1;                                                            0576
        [vcvtp]_done;                                                      0577
        BUMP vcvtp;                                                        0578
        !POP s,r1;                                                         0579
        END;                                                               0580
      RETURN (FALSE,loc);                                                  0581
      END.                                                                 0582
FINISH                                                                     0104
%...Merge, Update and friends...%                                          0816
   MOVED AFTER FINISH TO SAVE SPACE IN THE SYSTEM                          0891
   (mergbranch)PROC(bug1,bug2);                                            0787
      LOCAL stid1,stid2,stid3,stid4,zerw,stidx,rlevcnt,kpr;                0788
      REF kpr;                                                             0789
      zerw_0;                                                              0790
      &kpr_kprget();                                                       0797
      stidx_bug1;                                                          0799
      rlevcnt_levdown;                                                     0798
      IF (stid1_getsub(bug1)) = bug1 THEN stid1_endfil                     0793
          ELSE stid1_plxset(stid1 : stid2);                                0795
      IF (stid3_getsub(bug2)) = bug2 THEN RETURN;                          0794
      stid3 _ plxset(stid3 : stid4);                                       0796
      WHILE merge ($stidx,$stid1,&kpr,110,FALSE) # -2 DO NULL;             0791
      RETURN; END.                                                         0792
   (merggrp)PROC(bug1,bug2,bug3,bug4);                                     0767
      LOCAL zerw,stidx,rlevcnt,kpr; %BEWARE bug1 to zerw are sequenced%    0768
      REF kpr;                                                             0773
      zerw_0;                                                              0769
      bug1_grptst(bug1,bug2 : bug2);                                       0770
      bug3_grptst(bug3,bug4 : bug4);                                       0781
      stidx_colgdest(bug1 : rlevcnt);                                      0771
      &kpr _ kprget();                                                     0772
      WHILE merge($stidx,$bug1,&kpr,110,FALSE) # -2 DO NULL;               0774
      RETURN; END.                                                         0775
   (mergplx)PROC(bug1,bug2);                                               0776
      LOCAL stid1,stid2,stid3,stid4,zerw,stidx,rlevcnt,kpr;                0777
      REF kpr;                                                             0778
      zerw_0;                                                              0779
      &kpr_kprget();                                                       0786
      stid1_plxset(bug1 : stid2);                                          0780
      stidx_getup(stid1); rlevcnt_levdown;                                 0782
      stid3_plxset(bug2: stid4);                                           0783
      WHILE merge ($stidx,$stid1,&kpr,110,FALSE) # -2 DO NULL;             0784
      RETURN; END.                                                         0785
   (merge)PROC(stidx,inlst,kpr,percnt,copyf);                              0461
      LOCAL rlevcnt,ngrp,cxxb,n,stidn,cdtbm,crngm,cdtbloc,crngloc;         0462
      REF stidx,cxxb,cdtbloc,crngloc;                                      0463
```

```
   %map out nls pages%                                                    0843
      hmapout();                                                          0844
   &keypr_kpr;                                                            0464
   rlevcnt_stidx[1];                                                      0465
   &cxxb_IF copyf THEN $ccopgro ELSE $cmovgro;                            0466
   mersiz_40B;                                                            0467
   merrff_FALSE;                                                          0468
   % set structure and date block bounds %                               0802
      cdtbm_dtbm*percnt/100;                                              0803
      crngm_rngm*percnt/100;                                              0804
      &cdtbloc_$dtbl+&crngloc_filehead[stidx.stfile]-$filhed;             0805
      &crngloc_&crngloc+$rngl;                                            0806
   n_inlst; ngrp_0;                                                       0469
   WHILE [n] # 0 DO BEGIN                                                 0470
      BUMP ngrp; n_n+2;                                                   0471
      END;                                                                0472
   LOOP BEGIN                                                             0473
      FOR n _ 0 UP UNTIL >= ngrp DO BEGIN                                 0474
         r1_$v; .r1left_-n; r2_v;                                         0475
         (lp): !EXCH r2,1(r1); !AOBJN r1,lp;                              0476
         lodkey (n,gostid(n+n+inlst));                                    0477
         merorder(n+1);                                                   0478
         END;                                                             0479
      %test for no input%                                                 0480
         IF [v-2] = endfil THEN BEGIN hmapin(); RETURN(-2); END;          0481
      LOOP BEGIN                                                          0482
         IF cdtbloc>=cdtbm OR crngloc>=crngm THEN BEGIN hmapin();
         RETURN(-1); END;                                                 0483
            %this is turned off until outfull is written%                 0484
         IF merrff THEN BEGIN                                             0485
            mersiz_mersiz+40B;                                            0486
            merrff_ FALSE;                                                0487
            EXIT;                                                         0488
            END;                                                          0489
         n_ (v-$vcv)/mersiz;                                              0490
         stidn_g3nstid(n+n+inlst);                                        0491
         stidx _cxxb (stidx, rlevcnt, [v-2], [v-2], FALSE, 0);            0492
         rlevcnt _ levsuc;                                                0493
         IF NOT lodkey (n, stidn) THEN BEGIN hmapin(); RETURN(n);
         END;                                                             0494
         merorder(ngrp);                                                  0495
         END;                                                             0496
      END;                                                                0497
   END.                                                                   0498
(merorder)PROC(n);                                                        0437
   LOCAL i;                                                               0438
   FOR i _ 1 UP UNTIL>= n DO BEGIN                                        0439
      r2 _ v[i]; r1_v[i-1];                                               0440
      IF cmmpare() # 0 THEN RETURN;                                       0441
      v[i] _ v[i-1] := v[i];                                              0442
      END;                                                                0443
   RETURN; END.                                                           0444
(cmmpare)PROC;                                                            0445
   !HLL r1,-1(r1); !HLL r2,-1(r2);                                        0446
   (cmma): !MOVE r3,0(r1); !CAMN r3,0(r2); GOTO cmme;                     0447
   IF SKIP !CAMG r3,0(r2) THEN RETURN(FALSE);                             0448
```

```
   RETURN(TRUE);                                                          0449
   (cmme): !AOBJP r1,cmmx;                                                0450
   (cmmf): !AOBJN r2,cmma;                                                0451
   IF keydone($r2) THEN RETURN (FALSE);                                   0452
   GOTO cmma;                                                             0453
   (cmmx): IF keydone($r1) THEN BEGIN                                     0454
      !AOBJN r2,cmmxa;                                                    0455
      IF keydone($r2) THEN RETURN(-1);                                    0456
      (cmmxa): RETURN(TRUE);                                              0457
      END;                                                                0458
   GOTO cmmf;                                                             0459
   END.                                                                   0460
(keydone)PROC(loc);                                                       0499
   LOCAL r,wdcnt,sloc,n,vcvl;                                             0500
   REF loc;                                                               0501
   vcvl_$vcv;                                                             0502
   IF loc < vcvl THEN vcvl_$vcvu;                                         0503
   n _ (loc-vcvl)/mersiz;                                                 0504
   r _ vcvl + n*mersiz;                                                   0505
   IF [r+1] .A 7777B = 0 THEN BEGIN                                       0506
      wdcnt _ (sloc_loc)-r+11;                                            0507
      IF wdcnt+3 > mersiz THEN BEGIN                                      0508
         merrff_ TRUE;                                                    0509
         RETURN (TRUE);                                                   0510
         END;                                                             0511
      !PUSH s,r1;                                                         0512
      !PUSH s,r2;                                                         0513
      r1 _ keypr ([r], r+2, wdcnt : wdcnt);                               0514
      .r1left _ -wdcnt;                                                   0515
      [r+1] _ r1;                                                         0516
      r1 _ sloc .A 777777B;                                               0517
      .r1left _ r1-wdcnt - 2 - r;                                         0518
      a1_r1;                                                              0519
      !POP s,r2;                                                          0520
      !POP s,r1;                                                          0521
      loc_a1;                                                             0522
      RETURN (FALSE);                                                     0523
      END;                                                                0524
   RETURN (TRUE);                                                         0525
   END.                                                                   0526
(update)PROC(stidx,inlst,kpr,udpr,percnt,initf);                          0629
   LOCAL ngrp,n,dhold,cdtbm,crngm,cdtbloc,crngloc;                        0630
   REF udpr,initf,cdtbloc,crngloc;                                        0631
   &keypr_kpr;                                                            0632
   mersiz_40B;                                                            0633
   merrff_FALSE;                                                          0634
   % set structure and date block bounds %                               0807
      cdtbm_dtbm*percnt/100;                                              0808
      crngm_rngm*percnt/100;                                              0809
      &cdtbloc_$dtbl+&crngloc_filehead[[stidx].stfile]-$filhed;           0810
      &crngloc_&crngloc+$rngl;                                            0811
   IF initf THEN BEGIN                                                    0635
      initf_dhold_FALSE;                                                  0636
      &vcvulp_$vcvul;                                                     0637
      vcvulp_0;                                                           0638
      END;                                                                0639
```

```
   LOOP BEGIN                                                          0640
      n_inlst; ngrp_0;                                                 0641
      WHILE [n] # 0 DO BEGIN                                           0642
         BUMP ngrp; n_n+2;                                             0643
         END;                                                          0644
      FOR n _ 0 UP UNTIL >= ngrp DO BEGIN                              0645
         r1_$v; .r1left _-n; r2_v;                                     0646
         (lpa): !EXCH r2,1(r1); !AOBJN r1,lpa;                         0647
         lodkey (n,gostid(n+n+inlst));                                 0648
         merorder(n+1);                                                0649
         END;                                                          0650
      LOOP BEGIN                                                       0651
         IF merrff THEN BEGIN %reformat key area%                      0652
            mersiz_mersiz+40B;                                         0653
            merrff_FALSE;                                              0654
            EXIT;                                                      0655
            END;                                                       0656
         IF cdtbloc>=cdtbm OR crngloc>=crngm THEN RETURN(-1);          0812
         r1 _ $vcvu+2; r2_v;                                           0657
         IF dhold THEN                                                 0658
         CASE cmmpare() OF                                             0659
            =1: BEGIN %send list%                                      0660
               CALL udpr($vcvul, stidx);                               0661
               &vcvulp_$vcvul;                                         0662
               dhold_FALSE;                                            0663
               END;                                                    0664
            =0:  NULL; %error, file out of order%                      0665
            ENDCASE;                                                   0666
         IF  [v-2] = endfil THEN RETURN (-2);                          0667
         IF NOT dhold THEN BEGIN %copy to hold value%                  0668
            r1 _ [v-1]; r2_ -(.r1left) + 2; r2 _ r2 .A 777777B;        0669
            r1_$vcvu; .r1left _ v-2; !BLT r1,vcvu(r2);                 0670
            dhold_TRUE;                                                0671
            END;                                                       0672
         %add item to list%                                            0673
            vcvulp _ [v-2];                                            0674
            BUMP &vcvulp;                                              0675
            vcvulp _ n _ (v-$vcv)/mersiz;                              0676
            BUMP &vcvulp;                                              0677
            vcvulp _ 0;                                                0678
         %get new item%                                                0679
            IF NOT lodkey (n,g3nstid (n+n+inlst)) THEN RETURN (n,
            $vcvul);                                                   0680
            merorder(ngrp);                                            0681
         END;                                                          0682
      END;                                                             0683
   END.                                                                0684
(gostid)PROC(loc);                                                     0709
   LOCAL lb;                                                           0710
   REF loc,lb;                                                         0711
   &lb_&loc+1;                                                         0712
   IF loc#endfil THEN loc_grptst(loc,lb:lb);                           0713
   RETURN (loc);                                                       0714
   END.                                                                0715
(g3nstid)PROC(loc);                                                    0716
   LOCAL lb;                                                           0717
```

```
      REF loc,lb;                                                    0718
      &lb_&loc+1;                                                    0719
      IF loc # endfil THEN BEGIN                                     0720
         IF loc = lb THEN loc _ endfil ELSE loc _ getsuc(loc);       0721
         END;                                                        0722
      RETURN (loc);                                                  0723
      END.                                                           0724
%old collector-sorter%                                               0890
   (colsrt)PROC;                                                     0113
      LOCAL count, stmtfg;                                           0119
      STATE _ colstate, spec;                                        0114
      LOOP BEGIN                                                     0115
         crlf();                                                     0116
         count _ 0;                                                  0117
         WHILE (count _ count+1) <= fedind DO todco(SP);             0118
         todco('-);                                                  0120
         echoff();                                                   0121
         CASE inpcuc() OF                                            0122
            ='D: BEGIN echo($"Delete Keys? ");                       0123
               strpkey _ answer() END;                               0124
            ='E: BEGIN echo($"Execute Quit");                        0272
               getctc(); reset() END;                                0273
            ='F: BEGIN echo($"File List ");                          0125
               IF stmtfg _ (lookc() # SP) THEN BEGIN %read names from a
               statement%                                            0262
                  tbug($b1);                                         0263
                  CCPOS SF(b1);                                      0264
               END ELSE input();                                     0265
               csetif(stmtfg); END;                                  0127
            ='G: BEGIN echo($"Go ");                                 0128
               getctc();                                             0129
               chkprm();                                             0130
               ccontrol(&tda); END;                                  0131
            ='I: BEGIN echo($"Initialise ");                         0132
               getctc();                                             0133
               ninfil _ 0; *outnam* _ NULL;                          0134
               strpkey _ lngflg _ sortfg _ FALSE;                    0135
               smtmax _ 12000; END;                                  0136
            = 'L: BEGIN echo($"Length Key ?");                       0137
               lngflg _ answer() END;                                0138
            ='M: BEGIN echo($"Max Number Stmts/outtput file =");     0139
            smtmax _ gttnum(smtmax) END;                             0140
         ='O: BEGIN echo ($"Output File Name: ");                    0141
            coutnam() END;                                           0143
         ='S: BEGIN echo($"Sort ?");                                 0144
            sortfg _ answer() END;                                   0145
         ='V: tv();                                                  0146
      ENDCASE error();                                               0147
      END;                                                           0148
      END.                                                           0149
   (chkprm)PROC;                                                     0150
      IF outnam.L = 0 THEN err($"No Output File Name");              0151
      IF ninfil = 0 THEN err($"No Input Files");                     0152
      RETURN END.                                                    0153
   (csetif)PROC(stmtfg);                                             0154
      %READ list of file nams from keyboard fro now%                 0155
```

```
   LOCAL infastr, maxchr, chrused, strlng, char; REF infastr;     0156
   ninfil _ chrused _ 0;                                          0157
   maxchr _ 100*5;                                                0158
   &infastr _ $nambuf;                                            0159
   IF NOT stmtfg THEN echon();                                    0190
   LOOP BEGIN                                                     0160
      IF ninfil = 50 THEN err($"Too Many Input File Names");      0161
      maxchr _ maxchr-5; %room o for astring control word%        0162
      infastr.LH _ maxchr-chrused;                                0163
      *infastr* _ NULL;                                           0164
      IF stmtfg THEN BEGIN                                        0266
         WHILE (char _ READC) = SP DO NULL;                       0267
         IF char = ENDCHR THEN EXIT;                              0268
         WHILE (char # ',) AND (char # SP) AND (char # ENDCHR) DO
         BEGIN *infastr* _ *infastr*, char; char _ READC END;     0269
      END ELSE BEGIN                                              0270
         crlf();                                                  0165
         IF curchr = CA THEN EXIT;                                0191
         IF rdlit(&infastr, $rnmdel) = ovrflo THEN err($"Name space
         exhausted");                                             0166
      END;                                                        0271
      namist[ninfil := ninfil+1] _ &infastr;                      0167
      &infastr _ &infastr+ (strlng _ infastr.L+4)/5+1; %address of
      next room%                                                  0168
      chrused _ chrused+strlng+5;                                 0169
   END;                                                           0170
   namist[ninfil] _ -1; %end marker%                              0171
   RETURN END.                                                    0172
(coutnam)PROC;                                                    0173
   %read outfile name%                                            0174
      *outnam* _ NULL;                                            0175
      echon();                                                    0192
      crlf();                                                     0176
      IF rdlit($outnam, $rnmdel) = ovrflo THEN err($"Name Too Long");
                                                                  0177
      cversion _ 1;                                               0178
      RETURN END.                                                 0179
(ccmpare)PROC(i1,i2);                                               02
   LOCAL ccval;                                                   0346
   %Use p1 through p4 globally%                                     03
   IF (ccval_vcv[i2]-vcv[i1])>0                                   0347
      THEN RETURN (TRUE)                                          0348
      ELSE IF ccval<0 THEN RETURN (FALSE);                        0349
   IF vcv[i1] = 0 THEN RETURN (FALSE);                            0350
   ccval_v[i1]; CCPOS SF(ccval); FIND ^p1;                          04
   ccval_v[i2]; CCPOS SF(ccval); FIND ^p3;                          05
   LOOP BEGIN                                                       06
      IF NOT setkey($p1, $p2) THEN RETURN(TRUE);                    07
      IF NOT setkey($p3, $p4) THEN RETURN(FALSE);                   08
      *stn* _ p2 p1;                                                09
      *lit* _ p4 p3;                                               010
      CASE cmpstr($stn, $lit) OF                                   011
         =less: RETURN(TRUE);                                      012
         =gtr: RETURN(FALSE);                                      013
         ENDCASE;                                                  014
      END;                                                         015
```

```
      END.                                                          016
(setkey)PROC(ptr1, ptr2);                                           017
   REF ptr1, ptr2;                                                  018
   CCPOS ptr1;                                                      019
   RETURN(FIND '@ ^ptr2 [~'@] ^ptr1 _ptr1, ptr1, ptr1[1]);          020
   END.                                                             021
(cccval) PROCEDURE(stid);                                          0352
   LOCAL ps,char,val,cnt;                                          0353
   CCPOS SF(stid); FIND ^p1;                                       0354
   IF NOT setkey ($p1, $p2) THEN RETURN (0);                       0355
   ps_chif+$val;                                                   0356
   cnt_val_0;                                                      0357
   IF lngflg THEN BEGIN                                            0358
      ^ps _ MIN (77B, p1[1] - p2[1]);                              0359
      BUMP cnt;                                                    0360
      END;                                                         0361
   CCPOS p2;                                                       0362
   WHILE cnt < 5 AND (char_READC) # '@ DO BEGIN                    0363
      ^ps_char;                                                    0364
      BUMP cnt;                                                    0365
      END;                                                         0366
   RETURN (val); END.                                              0367
(addrec)PROC(stid);                                                 022
   %add stid into vector%                                           023
   LOCAL index, index1;                                             024
   IF (index _ vtop _ vtop + 1) > vmax THEN creror($"Too Many Input
   statements");                                                    025
   v[vtop] _ stid;                                                  026
   vcv[vtop] _ cccval(stid);                                       0368
   WHILE (index1 _ index / 2) > 0 DO                                027
      IF NOT  ccmpare(index, index1) THEN BEGIN                     028
         vcv[index1] _ vcv[index] := vcv[index1];                  0371
         v[index1] _ v[index := index1] := v[index1]               0369
         END ELSE EXIT;                                           0370
   RETURN END.                                                      029
(gnstid)PROC;                                                       030
   %Return next stid in sequence%                                   031
   LOCAL stid;                                                     0275
   (gnstdloop): IF NOT infopn THEN IF NOT opninf() THEN
   RETURN(endfil);                                                  032
   IF (stid _ seqgen(&colsw)) = endfil THEN BEGIN                   033
      closeseq(&colsw);                                            0380
      close(colda.dacsp.stfile :=0);                               034
      infopn _ 0;                                                  035
      GOTO gnstdloop END;                                          036
   RETURN(stid) END.
                                                                   037
(opstmt)PROC(stid, s4ort);                                         038
   %copy next input statemnt to file%                               039
   LOCAL TEXT POINTER z1, z2;                                      0872
   IF outcnt >= smtmax THEN BEGIN                                   040
      %file full...open next one%                                   041
      IF NOT s4ort THEN IF cversion > 1 THEN closeu(cstid.stfile);
                                                                   0248
      cstid.stfile _ copen(cversion := cversion+1, s4ort);          042
      cstid.stpsid _ origin;                                        043
```

```
      outcnt _ 0;                                                    091
      END;                                                           044
   FIND SF(stid) ^z1 SE(stid) ^z2;                                  0870
   cstid _IF stid.stastr THEN                                        045
      cinssta(cstid, levsuc, $z1, $z2)                              0867
   ELSE ccopsta(cstid, levsuc, stid, FALSE, 0);                     0871
   BUMP outcnt;                                                      046
   IF s4ort THEN addrec(cstid);                                      047
   RETURN END.                                                       048
(ccontrol)PROC(da);                                                  049
   LOCAL stid;                                                       051
   REF da;                                                           050
   &colda _ &da;                                                     052
   initcl(); %initialise things%                                     053
   WHILE (stid _ gnstid()) # endfil DO opstmt(stid, sortfg);         054
   crlf();                                                          0198
   typeas($"Input Finished");                                        056
   IF sortfg THEN finsort();                                         055
   cwrpup();                                                         057
   reset() END.
                                                                     058
% (cmpstr) has been moved to (utilty,cmpstr) %                      0838
(opninf)PROC;                                                        072
   %1--open nexxt input file in list                                 073
   2-- set up seqwrk to read in first psid (not origin)              074
   3----Retturn TRUE if ok, false if error or no more files%         075
   LOCAL wrkstr, colstd, sv1, sv2;                                  0873
   REF wrkstr;                                                      0874
   &wrkstr _ getstring(1000, $dspblk);                              0875
   ON SIGNAL ELSE freestring(&wrkstr, $dspblk);                     0876
   (opninloop):                                                      076
   IF namlst[namndx _ namndx+1] <= 0 THEN                           0877
      BEGIN                                                         0879
      freestring(&wrkstr, $dspblk);                                 0881
      RETURN(FALSE); %no more files%                                0878
      END;                                                          0880
   crlf();                                                           077
   cpysr(namlst[namndx], &wrkstr); %copy into wrkstr for openning%
                                                                    0193
   IF NOT colda.dacsp.stfile _  open(cgtjfn(&wrkstr), &wrkstr) THEN
                                                                     078
      BEGIN                                                         0882
      *wrkstr* _ *wrkstr*, "--Open Fail";                           0194
      typeas(&wrkstr);                                               079
      GOTO opninloop                                                 080
      END                                                           0883
   ELSE colda.dacsp.stpsid _ origin;                                0884
   typeas(&wrkstr);                                                 0195
   colda.dacsp_getsub(colda.dacsp);                                 0381
   &colsw_openseq(colstd _ colda.dacsp, seqend(colstd, sv1 _
   colda.davspec, sv2 _ colda.davspc2),sv1,                         0382
      sv2,colda.dausqcod,colda.dacacode);                           0383
   infopn _ TRUE;                                                   0822
   freestring(&wrkstr, $dspblk);                                    0885
   RETURN END.
                                                                     084
```

```
(cgtjfn)PROC(astrng);                                                  0205
   %RETURN jfn of file indicated by astrng%                            0206
   LOCAL jfn;                                                          0207
   IF NOT (jfn _ lgetjfn (0, astrng, $nlsext, gtjoif, astrng)) THEN
                                                                       0208
      err (astrng);                                                    0814
   RETURN(jfn) END.                                                    0209
(initcl)PROC;                                                          085
   vtop _ infopn _ 0;                                                  086
   cversion _ 1;                                                       0260
   namndx _ -1;                                                        0196
   outcnt _ smtmax+1;                                                  088
   IF [flntadr(colda.dacsp.stfile)].flexis THEN
   close(colda.dacsp.stfile := 0);                                     089
   RETURN END.
                                                                       090
(copen)PROC(version, s4ort);                                           0180
   %Open next output file, and return NLS file numbeer%                0181
   LOCAL fileno, jfn, flgbits, nfl;                                    0182
   IF s4ort THEN *lit* _ "xxxsort", version+60B ELSE                   0259
   *lit* _ *outnam*, version+60B;                                      0183
   IF NOT (jfn _ lgetjfn (0, $lit, $nlsext, getgtjflg(write, origff,
   oldvrsn), $lit)) THEN creror($"Output File Open Fail");             0184
   jfnstr(jfn, $lit);                                                  0185
   fileno _ addfile(jfn, $lit);                                        0186
   %now do dummy open and close to create it%                          0219
      IF NOT sysopen(jfn, write, random, $lit) THEN err($lit);         0220
      IF NOT sysclose(jfn .V 4B11, $lit) THEN err($lit);               0223
   IF NOT opnfil(fileno, $lit) THEN err($lit);                         0225
   IF lit.L > empty THEN                                               0860
      BEGIN                                                            0861
      dismes (2, $lit);                                                0862
      <AUXCOD, pause> (2000);                                          0863
      END;                                                             0864
   resetf(fileno); %reset file%                                        0261
   RETURN(fileno) END.                                                 0188
(creror)PROC(errcod);                                                  092
   nlmode _ typewriter;                                                0215
   IF errcod THEN typeas(errcod);                                      093
   cwrpup();                                                           094
   err($"Fatal Error In COLSORT");                                     095
   END.
                                                                       096
(cwrpup)PROC;                                                          097
    closeu(cstid.stfile := 0);                                         0274
   closeall();                                                         099
   *outnam* _ *outnam*, '1; %open and load first output file%          0226
   colda.dacsp.stfile _ open(cgtjfn($outnam), $outnam);                0203
   colda.dacsp.stpsid _ origin;                                        0204
   RETURN END.                                                         0100
(tinsort)PROC;                                                         0101
   LOCAL i, fflag, oldtime, timeused;                                  0227
   oldtime _ 0;                                                        0285
   i _ vtop+1;                                                         0228
   WHILE (i_i-1) >= 2 DO BEGIN                                         0229
      r1 _4B11; !JSYS runtm; timeused _ r1/r2;                         0277
```

```
      IF timeused - oldtime > 60 THEN BEGIN %type out a message%  0278
         oldtime _ timeused;                                      0279
         crlf(); typeas($"Sort Running, time = ");                0280
         r1 _ 101B; r2 _ oldtime; r3 _ 10; IF NOT SKIP !JSYS nout
         THEN err(0);                                             0281
         typeas($" Seconds, node = ");                            0282
         r1 _ 101B; r2 _ vtop-i+1; r3 _ 10; IF NOT SKIP !JSYS nout
         THEN err(0);                                             0283
         END;                                                     0284
      siftup(1,i);                                                0230
      v[1] _ v[i] := v[1];                                        0373
      vcv[1] _ vcv[i] := vcv[1];                                  0378
      END;                                                        0232
   %now put files in order%                                       0233
      cversion _ 1;                                               0249
      i _ 0;                                                      0250
      outcnt _ smtmax+1;                                          0252
      WHILE (i _ i+1) <= vtop DO BEGIN opstmt(v[i], FALSE); IF
      strpkey THEN BEGIN CCPOS SF(cstid);                         0251
         FIND ^p1;                                                0253
         fflag _ FALSE;                                           0257
         WHILE setkey($p1, $p2) DO fflag _ TRUE;                  0254
         IF fflag THEN ST p1 _ p2 SE(p2);                         0255
      END;                                                        0256
      END;                                                        0258
      %delete keys here%                                          0236
   RETURN END.
                                                                  0103
(siftup)PROC(i,n);                                                0237
   %sort tree into vector...from floydd%                          0238
   LOCAL j;                                                       0239
   v_v[i]; vcv_vcv[i];                                            0240
   LOOP                                                           0241
      IF (j _ i+i) <= n THEN BEGIN                                0242
         IF j<n THEN IF  ccmpare(j, j+1) THEN BUMP j;             0243
         IF NOT ccmpare(j, 0) THEN BEGIN                          0244
            vcv[i] _ vcv[j];                                      0377
            v[i:=j] _ v[j]                                        0375
            END ELSE EXIT;                                        0376
      END ELSE EXIT;                                              0245
   v[i] _ v; vcv[i] _ vcv;                                        0379
   RETURN END.                                                    0247
```

CORE SUPPORT

```
<NLS>CORESUPPORT.NLS;1, 26-NOV-73 21:49 CHI ;                          02
FILE coresupport   % L10 to <rel-nls>coresupport %                     03
% inptbk %                                                            04
   (chrpos) PROCEDURE (char, astrng); %scan string for character%     05
      %chr in first parameter, a-string addr in second%                06
      %return TRUE if char in string, FALSE otherwise.%                07
      %-----------------%                                              08
      LOCAL tmpchr;                                                    09
      REF astrng;                                                     010
      CCPOS *astrng*;                                                 011
      LOOP                                                            012
         IF (tmpchr _ READC) = char THEN RETURN(TRUE)                 013
         ELSE IF tmpchr = ENDCHR THEN RETURN(FALSE);                  014
      END.                                                            015
                                                                      016
   (clrbuf) %clear the todas input and (optionally) output%           017
      %if outflg is non-zero the output buffer is also cleared.       018
      nothing is  returned%                                           019
      %-----------------%                                             020
      PROCEDURE (outflg);                                             021
         IF outflg THEN                                               022
            BEGIN                                                     023
            %jsys to clear output buffer%                             024
               r1 _ 101B;                                             025
               !JSYS cfobf;                                           026
            END;                                                      027
         %jsys to clear input buffer%                                 028
            r1 _ 100B;                                                029
            !JSYS cfibf;                                              030
         buffn _ buffs; % reset input buffer pointers %               031
         RETURN;                                                      032
         END.                                                         033
                                                                     0475
FINISH
```

CPROGRAMS

```
< NLS, CPROGRAMS.NLS.20, >, 7-Dec-77 19:50 HGL ;;;;% invoke the TENLDR
saved in our address space to load a rel file
  the file is loaded into the user program buffer, and the tenldr
expects three arguments set up %
FILE cprograms % L10 <rel-nls>CPROGRAMS %% (110,)                    01442
(rel-nls,CPROGRAMS.rel,) %                                           06088
DECLARE EXTERNAL cprostart;                                          03877

DECLARE symloc=116B, nlsloader=117B, ddtsptr=770001B;
REGISTER r1 = 1, r2 = 2;  REF tda, symloc, nlsloader, ddtsptr;       03876
                                                                     03878
%compile%
   (cpconan)    % content analyzer pattern compile routine %         03879
      PROCEDURE(                                                     03880
      % FORMAL ARGUMENTS %                                          03881
         tptr,  % ptr to content pattern string %                    03882
         da);   % ptr to display area %                             03883
                                                                     03884
      % RETURNS %                                                   03885
         % 1) address of the program in the programs buffer %       03886
      REF tptr, da;                                                  03887
      LOCAL TEXT POINTER tptr2;                                      03888
      LOCAL STRING patternstr[500];                                  03889
      % -------------- %                                            03890
      *patternstr* _ tptr SE(tptr), " ;;";                           03891
      FIND SF(*patternstr*) ^tptr2;                                  03892
      RETURN (gpconan( $tptr2, &da ));                               03893
      END.                                                           03894

   (cpcmpfl)    %Core NLS Compile File command %                     03895
      PROCEDURE                                                      03896
         (type,
            % FALSE > cmpnam is adr of parsed link data structure %  03897

               % also file name associated with link must have
               extenstion name = *savext* %                          03898
            % TRUE > cmpnam is adr of compiler name string %         03899
                                                                     03900
         cmpnam,
         ofilnam,       % string containing the name of the output file
                                                                     03901
         %                                                           03902
         da);   % display area descriptor %                         03903
                                                                     03904
      LOCAL jfn, errors, cjfn;                                       03905
      LOCAL TEXT POINTER tp;                                         03906
      LOCAL STRING locstr[40];                                       03907
      REF cmpnam, ofilnam, da;                                       03908
                                                                     03909
      % open the output file %
         IF NOT jfn _ lgetjfn (0, &ofilnam, $relext, gtjoosf, $lit) THEN
                                                                     03910
                                                                     03911
            SIGNAL (ofilerr, $lit);
         IF NOT sysopen (jfn, write, bintyp, $lit) THEN              03912
                                                                     03913
            BEGIN                                                    03914
            reljfn (jfn);                                            03915
            SIGNAL (ofilerr, $lit);                                  03916
            END;
      % get jfn for compiler if passed parsed link data structure %
```

```
                                                                    03917
      cjfn _ 0;                                                     03918
      IF NOT type THEN                                              03919
         IF NOT (cjfn _ upgjfn( &cmpnam ) ) THEN                    03920
            SIGNAL( prcerr, $lit)                                   03921
         ELSE                                                       03922
            BEGIN                                                   03923
            jfntostr( cjfn, $locstr, 000100B6);                     03924
            IF *locstr* # *savext* THEN                             03925
               SIGNAL( prcerr, $"Illegal Compiler Name")            03926
            END;                                                    03927
   tp[0] _ da.dacsp; tp[1] _ 1;                                     03928
   errors _                                                         03929
      processor( IF type THEN &cmpnam ELSE 0,                       03930
         $tp, &da, cmptyp, jfn, cjfn);                              03931
   % close output file (but keep jfn) %                             03932
      IF NOT sysclose (jfn .V 4B11, $lit) THEN                      03933
         BEGIN                                                      03934
         reljfn (jfn);                                              03935
         dismes (2, $lit);                                          03936
         GOTO STATE;                                                03937
         END;                                                       03938
   % delete old versions, if no errors and no rubout %              03939
      IF errors = 0 AND NOT inptrf THEN delovsrns (jfn, nvtk - 1);
                                                                    03940
      reljfn (jfn);                                                 03941
   RETURN (errors);                                                 03942
   END.                                                             03943
                                                                    03944
(cpcmproc)    PROCEDURE   % does Compile Procedure %                03945
   (stid, % stid where compilation is to start %                    03946
   da);   % display area descriptor %                               03947
                                                                    03948
   % Compile a procedure.  If the compilation is successful and there
   is room in the buffer for the code and room in the user program
   stack, add the program to the stack of user programs, and do a
   procedure replace.  Mark DDT's symbol table. %                   03949
                                                                    03950
   LOCAL size, errs;                                                03951
   LOCAL TEXT POINTER tp, tpb, tpe;                                 03952
   LOCAL STRING tempstr[100], locstr[100];                          03953
   REF cm1tmp, da;                                                  03954
                                                                    03955
   IF NOT (FIND SF(stid) ^tp ['('] $NP ^tpb $LD ^tpe) THEN          03956
      SIGNAL (upgerr, $"couldn't find procedure name");             03957
   *lit* _ tpb tpe;                                                 03958
   *locstr* _ + tpb tpe;                                            03959
   % setup to use special sequence %                                03960
      cm5tmp _ da.davspec;                                          03961
      cm6tmp _ da.davspc2;                                          03962
      cm3tmp _ da.dacacode;                                         03963
      cm4tmp _ da.dausqcod;                                         03964
      da.davspec.vsusqf _ da.davspec.vsbrof _ TRUE;                 03965
      da.dausqcod _ $cpseqg;                                        03966
      da.dacacode _ 0;                                              03967
      &cm1tmp _ 0;                                                  03968
```

```
      ON SIGNAL ELSE                                                    03969
         BEGIN                                                          03970
         da.davspec _ cm5tmp;                                           03971
         da.davspc2 _ cm6tmp;                                           03972
         da.dacacode _ cm3tmp;                                          03973
         da.dausqcod _ cm4tmp;                                          03974
         IF &cm1tmp THEN closeseq( &cm1tmp := 0 );                      03975
         END;                                                           03976
   marksym ($locstr);                                                   03977
   [upgffbuf] _ upgbend - upgffbuf;   % space left in buffer %          03978
   errs _ <SEQFIL, processor>                                           03979
      (IF exp THEN $x110nam ELSE $110nam, $tp, &da, upgtyp, upgffbuf,
      0  :size);                                                        03980
   % cleanup from above %                                              03981
      da.davspec _ cm5tmp;                                              03982
      da.davspc2 _ cm6tmp;                                              03983
      da.dacacode _ cm3tmp;                                             03984
      da.dausqcod _ cm4tmp;                                             03985
      IF &cm1tmp THEN closeseq( &cm1tmp := 0 );                         03986
      ON SIGNAL ELSE;                                                   03987
   IF errs > 0 THEN                                                     03988
      BEGIN                                                             03989
      popsym();                                                         03990
      RETURN (errs)                                                     03991
      END                                                               03992
   ELSE IF upgskix >= $upgsksz THEN                                     03993
      BEGIN                                                             03994
      popsym();                                                         03995
      SIGNAL (upgerr, $"no more room in user program stack")            03996
      END                                                               03997
   ELSE                                                                 03998
      % compilation was successful and there was room in the buffer
      and there is room in stack %                                      03999
      BEGIN                                                             04000
      upgstk [upgskix _ upgskix + 1] _ upgffbuf;                        04001
      upgffbuf _ upgffbuf + size;                                       04002
      *upgnms* _ *upgnms*, SP, *locstr*;                                04003
      % do procedure replace %                                         04004
         IF ddtlookup( $locstr, FALSE ) THEN                            04005
            BEGIN                                                       04006
            IF rplproc( $locstr, $locstr) THEN                          04007
               BEGIN                                                    04008
               *tempstr* _ "Procedure ", *locstr*, " Replaced";         04009
               dismes(2, $tempstr);                                     04010
               END;                                                     04011
            END;                                                        04012
      END;                                                              04013
   RETURN (0);                                                          04014
   END.                                                                 04015
                                                                        04016
% sequence generator program for compile procedure %                   04052
   (cpseqg) PROCEDURE (sw, which); % Compile Procedure Seq Generator
   %                                                                    04017
      LOCAL vpc;                                                        04018
      LOCAL STRING locstr[200];                                         04019
      REF cm1tmp, sw;                                                   04020
```

```
      CASE which OF                                                    04021
         =sqopn: % called at open %                                    04022
            BEGIN                                                      04023
            vpc _ cm5tmp;  vpc.vsbrof _ TRUE;                          04024
            cm2mp _ 0;                                                 04025
            &cm1tmp _                                                  04026
               openseq( sw.swcstid, sw.swlbstid, vpc, cm6tmp, cm4tmp,
               cm3tmp);                                                04027
            END;                                                       04028
         =sqgnxt: % called for next in seq -- fall through %           04029
            IF FALSE = (cm2mp := TRUE) THEN                            04030
               BEGIN                                                   04031
               *locstr* _ "PROGRAM ", *lit*;                           04032
               send( &sw, $locstr);                                    04033
               REPEAT CASE( sqgnxt );                                  04034
               END                                                     04035
            ELSE                                                       04036
               LOOP                                                    04037
                  BEGIN                                                04038
                  IF (sw.swstid _ seqgen(&cm1tmp) ) = endfil THEN
                                                                       04039
                     send( &sw, $"FINISH");                            04040
                  cpysw( &cm1tmp, &sw);                                04041
                  sport( &sw );                                        04042
                  END;                                                 04043
         =sqcls: % called at close %                                   04044
            BEGIN                                                      04045
            closeseq( &cm1tmp := 0 );                                  04046
            END;                                                       04047
         ENDCASE err ($"Illegal call to a seq gen program"); % called
         for any other purpose -- error %                              04048
      RETURN;                                                          04049
      END.                                                             04050
                                                                       04051
(dinstupg) PROCEDURE   %  Deinstitute a program %                      04981
   (pgmaddr);   % address of of the user program %                     04982
   % "deinstitute" the specified user program --                       04983
      find any references to it and delete them, but don't give back its
      space to the buffer -- thus it can "reinstituted" %              04984
   LOCAL da;                                                           04985
   REF da;                                                             04986
   FOR &da _ $dpyare UP dal UNTIL >= $dpyare + dacnt*dal DO            04987
      IF da.daexis THEN                                                04988
         BEGIN  % fields pointing to program to be popped are reset %
                                                                       04989
         IF da.dacacode = pgmaddr THEN da.dacacode _ 0;                04990
         IF da.dausqcod = pgmaddr THEN da.dausqcod _ 0;                04991
         IF da.daukeycod = pgmaddr THEN da.daukeycod _ 0;              04992
         END;                                                          04993
   RETURN;                                                             04994
   END.                                                                04995
                                                                       04996
(gpbsz)  PROCEDURE  % Goto User program Buffer Size Set %              02314
   (size);  % Number of pages to be in new buffer %                    02315
                                                                       02316
   % RETURNS %                                                        02317
```

```
    % returns false for invalid request or address of first new page
      allocated %                                                    02318
    LOCAL stadr;                                                     02319
                                                                     02320
    IF rfpmax - size < minfpages OR (512*size) < (upgffbuf-upgbuf+1)
                                                                     02321
      THEN RETURN (FALSE);                                           02322
    cioseall();                                                      02323
    upgbsz _ size;                                                   02324
    rfpmin _ size + 1;  % Index to first file core page %            02325
    stadr _ upgbend + 1;                                             02326
    upgbend _ upgbuf + 512*size - 1;                                 02327
    openall();                                                       02328
    RETURN (stadr);                                                  02329
    END.                                                             02330
                                                                     02331
(gpgmrst)  PROCEDURE;  % Goto User program reset %                   03004
    % reset the stack and all display area descriptor fields having to do
    with user programs %                                             03005
    %reset user programs buffer size to default value%               05462
    ON SIGNAL                                                        03007
      = upgerr: RETURN;                                              03008
      ELSE;                                                          03009
    nddtdisarm(); % because NDDT is now a user program which will be
    popped here! %                                                   05984
    jnlprog _ FALSE; %incase jnldel was loaded%                      06029
    LOOP gppop();                                                    03010
    RETURN;                                                          03011
    END.                                                             03012
                                                                     03013
(gpconan)  PROCEDURE   % does Goto Program Content analyzer %        01831
    (tpb,  % text pointer to string containing pattern %             01832
    da);      % display area %                                       01833
                                                                     01834
    % Compile a content analyzer pattern.  If the compilation is
    successful and there is room in the buffer for the code and room in
    the user program stack, add the pattern to the stack of user programs
    -- errors cause a upgerr SIGNAL to be generated.  DDT's symbol table
    is marked. %                                                     01835
                                                                     01836
    LOCAL size, errs, location;                                      01837
    LOCAL TEXT POINTER tpe;                                          01838
    REF tpb, da;                                                     01839
                                                                     01840
    marksym (0);  % mark the symbol table %                          01841
    [upgffbuf] _ upgbend - upgffbuf;   % space left in buffer %      01842
    errs _ <SEQFIL, processor>                                       01843
      (IF exp THEN $x110nam ELSE $110nam, &tpb, &da, upgtyp, upgffbuf, 0
      :size);                                                        01844
    IF errs # 0 THEN                                                 01845
      BEGIN                                                          01846
      popsym();                                                      01847
      SIGNAL (upgerr, errs)                                          01848
      END                                                            01849
    ELSE IF upgskix >= $upgsksz THEN                                 01850
      BEGIN                                                          01851
```

```
      popsym();                                                          01852
      SIGNAL (upgerr, $"no more room in user program stack")             01853
      END                                                                01854
   ELSE                                                                  01855
      % compilation was successful and there was room in the buffer and
      there is room in stack %                                           01856
      BEGIN                                                              01857
      location _ upgstk [upgskix _ upgskix + 1] _ upgffbuf;              01858
      upgffbuf _ upgffbuf + size;                                        01859
      BUMP upgcacnt;                                                     01860
      FIND > tpb $NP ^tpb $5 PT ^tpe;                                    01861
      *upgnms* _ *upgnms*, SP, "UP", STRING(upgcacnt), '!, tpb tpe;
                                                                         01862
      END;                                                               01863
   RETURN (location);                                                    01864
   END.                                                                  01865
                                                                         01866
(gpexpgm)  PROCEDURE   % does Goto Program Execute a program %          01892
   (str);   % string containing name or number of the user program %
                                                                         01893
   % control is transferred to specified program by means of a CALL0%
                                                                         01894
   LOCAL upgaddr;                                                        01895
   REF str;                                                              01896
   upgaddr _ upgcnv (&str);                                              01897
   IF NOT upgaddr > 0 THEN                                               01898
      SIGNAL (upgerr, $"no address for program");                        01899
   [upgaddr]();                                                          01900
   RETURN;                                                               01901
   END.                                                                  01902
                                                                         01903
(gpget)    PROCEDURE   % does Goto Programs Get Rel  File command %
                                                                         05767
   (adstr,   % address of parsed link data structure %                   05768
   shomes);  % If TRUE, do dismes's; FALSE, only error mssages form
   loader get out. %                                                     05769
   % invoke the TENLDR saved in our address space to load a rel file
     the file is loaded into the user program buffer, and the tenldr
   expects three arguments set up %                                      05770
   LOCAL saveprogaddr, savsymloc, jfn, errs, lc, loader, entry, ddtloc,
   oneflag, extflag, pgmfield, try, stname, i, ptype, dtstr[40];         05771
   LOCAL TEXT POINTER tps,tpe, tp1, tp2;                                 05772
   LOCAL STRING errstr[200];                                             05773
   LOCAL STRING recurse[200];                                            05774
   LOCAL STRING filename[39];                                            05775
   LOCAL STRING extname[39];                                             05776
   LOCAL STRING tempstr[50];                                             05777
   LOCAL STRING extension[10];                                           05778
   LOCAL STRING str[200];                                                05779
   REF adstr, loader;                                                    05780
   % initialize %                                                        05781
      oneflag _ TRUE;                                                    05782
   % get the file name specified by the parsed link data structure %
                                                                         05783
      CASE lnbfls( 0, &adstr, $str) OF                                   05784
         = lhostn:  NULL;                                                05785
```

```
      ENDCASE err($"Remote File Manipulations Not Implemented Yet");
                                                                    05786
% edit the name string to get just the file name %                  05787
   extflag _ FALSE; % set to TRUE by the FIND stmt if extension is
   given %                                                          05788
   FIND SF(*str*) ^tps ^tpe;                                        05789
   IF FIND tpe > ['< [']> / '<^F> / '<ESC>] ^tps THEN NULL          05790
   ELSE IF NOT FIND tpe > LD THEN                                   05791
      err($"Illegal Rel File Name");                                05792
   IF FIND tps > ['.] ^tpe _tpe THEN                                05793
      BEGIN                                                         05794
      IF FIND LD THEN extflag _ TRUE;                               05795
      END                                                           05796
   ELSE                                                             05797
      IF FIND tps > ['<^F> / '<ESC>] ^tpe THEN                      05798
         BEGIN                                                      05799
         IF FIND LD THEN extflag _ TRUE;                            05800
         END                                                        05801
      ELSE                                                          05802
         IF NOT FIND tps > $LD (EOL / ENDCHR) ^tpe THEN             05803
            err($"Illegal Rel File Name");                          05804
   *filename* _ tps tpe;                                            05805
% save away current state values %                                 05806
   savsymloc _ symloc;                                              05807
% get the jfn for the rel file %                                   05808
   % all of the following rather obtuse code attempts to figure out
   what file is to be loaded and what type it is.  It is not
   necessary for the user to specify the extension name, as this
   grungy code manages to find that out.  If no extension is given,
   then extensions are attempted in the following order: REL, CA, SK,
   SG, SUBSYS, CML, PROC-REP.  If this sequence cannot be satisfied
   in the default directory for links (for bugged links if no
   directory is specified) or in the connected directory (for typed
   in links if no directory is specified) then the PROGRAMS directory
   is used and the same sequence is repeated %                      05809
   *extension* _ "REL"; try _ 0;                                    05810
   CASE jfn _ lgetjfn(0,$str,$extension,1B11,$errstr) OF            05811
      =FALSE: % getjfn failed, try some recovery strategy %         05812
         BEGIN                                                      05813
         WHILE NOT extflag DO                                       05814
            BEGIN % try to figure out what extension user wants %
                                                                    05815
            CASE try _ try + 1 OF                                   05816
               =1: *extension* _ "CA";                              05817
               =2: *extension* _ "SK";                              05818
               =3: *extension* _ "SG";                              05819
               =4: *extension* _ "SUBSYS";                          05820
               =5: *extension* _ "CML";                             05821
               =6: *extension* _ "PROC-REP";                        05822
               ENDCASE EXIT LOOP;                                   05823
            REPEAT CASE;                                            05824
            END;                                                    05825
         % maybe try default user program directory %               05826
            IF (oneflag := FALSE) AND                               05827
               ((*str*[1] # '<) OR (adstr[ue+1] = adstr[us+1])) THEN
                                                                    05828
```

```
                BEGIN                                                   05829
                *str*  _ "<PROGRAMS>", tps SE(*str*);                   05830
                *extension* _ "REL"; try _ 0;                           05831
                REPEAT CASE;                                            05832
                END;                                                    05833
          err($errstr);                                                 05834
          END;                                                          05835
   ENDCASE;                                                             05836
% record usage of the program %                                        06085
   jfntostr( jfn, $filename, 011100000001B);                            06086
   otflerrec(7, 0, $filename);                                          06087
% get rid of any ^F or altmodes in filename %                          05837
   jfntostr( jfn, $filename, 001000B6);                                 05838
   jfntostr( jfn, $extname, 000100B6);                                  05839
% find out what type of program for future instituting %                05840
   ptype _ 0;                                                           05841
   IF *extname* = "CA" THEN ptype _ 1                                   05842
   ELSE                                                                 05843
      IF *extname* = "SK" THEN ptype _ 2                                05844
      ELSE                                                              05845
         IF *extname* = "SG" THEN ptype _ 3                             05846
         ELSE                                                           05847
            IF *extname* = "PROC-REP" THEN ptype _ 4                    05848
            ELSE                                                        05849
               IF *extname* = "CML" THEN ptype _ 5                      05850
               ELSE                                                     05851
                  IF *extname* = "SUBSYS" THEN ptype _ 6                05852
                  ELSE NULL;                                            05853
% open the file %                                                       05854
   IF NOT SKIP !openf (jfn, 4400002B5) THEN                             05855
      BEGIN                                                             05856
      IF NOT SKIP !rljfn( jfn ) THEN NULL;                              05857
      err( $"Open File Failed" );                                       05858
      END;                                                              05859
%adjust the user program buffer size if necessary to fit the program
being loaded%                                                           05860
   gpadjbsz(jfn);                                                       05861
% invoke the tenldr %                                                  05862
   IF shomes THEN dismes(1, $"Loading User Program");                   05863
   &loader _ nlsloader;                                                 05864
   errs _ loader(jfn, upgffbuf+1, upgbend :lc);                         05865
   IF shomes THEN dismes(0);                                            05866
% the tenldr returns an error count and the next address in the
buffer %                                                                05867
   IF NOT errs THEN % good return %                                     05868
      BEGIN                                                             05869
      % mark the block in the user program buffer %                     05870
         IF upgskix >= $upgsksz THEN                                    05871
            SIGNAL (upgerr, $"no more room in user program stack")
                                                                        05872
         ELSE                                                           05873
            % load was successful and there was room in the buffer
            and there is room in stack %                                05874
            BEGIN                                                       05875
            upgstk[upgskix _ upgskix + 1] _ upgffbuf;                   05876
            % set string into user program buffer %                    05877
```

```
            *upgnms* _ *upgnms*, SP, *filename*;                    05878
         END;                                                       05879
      % update ddt's alt I-1 to include program just loaded %       05880
       ddtloc _ ddtsptr; % get ptr to alt i -1 %                    05881
       [ddtloc] _ symloc; % stuff away new symloc %                 05882
      % mark the block in the symbol table  %                       05883
       ddmrk[ddmrkx] _ savsymloc.RH - 1;                            05884
       IF (ddmrkx _ ddmrkx + 1 ) > ddmrkm THEN                      05885
          BEGIN                                                     05886
          ddmrkx _ ddmrkx - 1;                                      05887
          err($"Mark stack overflow");                              05888
          END;                                                      05889
      % lookup file name and set entry instruction %                05890
       stname _                                                     05891
          IF ptype = 5 %CML% THEN                                   05892
             mrkglookup( $filename : entry )                        05893
          ELSE                                                      05894
             ddtlookup( $filename, FALSE : entry );                 05895
       IF stname  THEN                                              05896
          [upgffbuf] _ 254B9 + entry %set up entry inst %           05897
       ELSE                                                         05898
          CASE ptype OF                                             05899
             =0, =1, =2, =3:       % REL, CA, SK, SG %            05900
                IF shomes THEN dismes(2,$"WARNING -- no entry to
                program ");                                         05901
             =4:                   % PROC-REP %                    05902
                BEGIN                                               05903
                *tempstr* _ "WARNING -- No Procedure Named ",
                *filename*;                                         05904
                IF shomes THEN dismes(2,$tempstr);                  05905
                END;                                                05906
             =5:                   % CML %                         05907
                BEGIN                                               05908
                *tempstr* _ "WARNING -- No Subsystem Named ",
                *filename*;                                         05909
                IF shomes THEN dismes(2,$tempstr);                  05910
                END;                                                05911
             =6:                   % SUBSYS %                      05912
                IF shomes THEN dismes(2,$"Don't Execute via RUN
                PROGRAM Command
                Use GOTO SUBSYSTEM Command");                       05913
             ENDCASE;                                               05914
      % set new limit on buffer space %                            05915
        saveprogaddr _ upgffbuf := lc;                              05916
      % institute program if so indicated by extension %           05917
       CASE ptype OF                                                05918
          = 1: % content analyzer %                                 05919
             BEGIN                                                  05920
             IF shomes THEN dismes(2,$"Instituting User Program as
             a Content Analyzer");                                  05921
             [lda()].dacacode _ saveprogaddr;                       05922
             END;                                                   05923
          = 2: % sort key %                                         05924
             BEGIN                                                  05925
             IF shomes THEN dismes(2,$"Instituting User Program as
             a Sort Key Program");                                  05926
```

```
      [lda()].daukeycod _ saveprogaddr;                          05927
      END;                                                       05928
   = 3: % sequence generator %                                   05929
      BEGIN                                                      05930
      IF shomes THEN dismes(2,$"Instituting User Program as
      a Sequence Generator");                                    05931
      [lda()].dausqcod _ saveprogaddr;                           05932
      END;                                                       05933
   = 4: % procedure replace %                                    05934
      IF stname THEN                                             05935
         BEGIN                                                   05936
         IF rplproc( $filename, $filename) THEN                  05937
            *tempstr* _ "Procedure ", *filename*, "
            Replaced"                                            05938
         ELSE                                                    05939
            *tempstr* _ "Old Procedure ", *filename*, " Does
            Not Exist";                                          05940
         IF shomes THEN dismes(2, $tempstr);                     05941
         END;                                                    05942
   = 5:     % cml %                                             05943
      IF stname THEN                                             05944
         BEGIN                                                   05945
         upgstk[upgskix].LH _ entry;                             05946
         dfnsubsys( entry, gtctrlbits(entry, $nlssubs),
         $nlssubs );                                             05947
         dfnsubsys( entry, gtctrlbits(entry, $allsubs),
         $allsubs );                                             05948
         *tempstr* _ "Subsystem ", *filename*, " Now
         Available (Attached)";                                  05949
         IF shomes THEN dismes(2, $tempstr);                     05950
         END;                                                    05951
   = 6:     % subsys %                                          05952
      BEGIN                                                      05953
      *tempstr* _ *filename*, ".CML";                            05954
      FOR i _ 0 UP UNTIL = 40 DO dtstr[i] _ adstr[i];            05955
      FIND SF(*tempstr*) ^tp1 SE(*tempstr*) ^tp2;                05956
      dtstr[fs] _ tp1; dtstr[fs+1] _ tp1[1];                     05957
      dtstr[fe] _ tp2; dtstr[fe+1] _ tp2[1];                     05958
      ON SIGNAL ELSE                                             05959
         BEGIN                                                   05960
         ON SIGNAL ELSE;                                         05961
         err($"Can't load frontend for this SUBSYS");            05962
         END;                                                    05963
      gpget($dtstr, shomes);                                     05964
      ON SIGNAL ELSE;                                            05965
      %upgstk[upgskix-1].LH _ upgstk[upgskix].LH;%               05966
      %mark the LH of the second from the top entry of the
      user program stack with -1 . This indicates that is a
      user .subsys program which has had a corresponding
      user .cml program automatically loaded on top of it.
      This flag is used by gppop so that the command Delete
      Last (program in buffer) will actually delete both the
      CML and Subsys portions of the user subsystem%             05967
      upgstk[upgskix-1].LH _ -1;                                 05968
      %BUMP DOWN upgskix;%                                       05969
      %popsym(); This has been removed so that detach
```

```
                subsystem will work properly. This implies that when
                loading a subsystem both the CML and the SUBSYS
                symbols will be on the symbol stack. These two blocks
                in the symbol table should therefore have distinct
                names (distinct names on the file statements)%    05970
                FIND SE(*upgnms*) $PT $NP ^tp2;                   05971
                *upgnms* _ SF(*upgnms*) tp2;                      05972
                END;                                              05973
             ENDCASE;                                             05974
          END                                                     05975
       ELSE                                                       05976
          BEGIN                                                   05977
          symloc _ savsymloc; % reset symloc as it may be clobbered by an
          error in loading %                                      05978
          err( $"Error in Loading");                              05979
          END;                                                    05980
   RETURN;                                                        05981
   END.                                                           05982
                                                                  05983
(getuprog) PROCEDURE                                              06030
   %FORMAL PARAMETERS%                                            06031
      (stptr);  %ptr to a string containg program name (or a directory
      name followed by a comma follwed by the file name.)%        06032
   % This procedure calls gpget with a flag saying not to put out
   program loading messages except for errors. %                  06033
   LOCAL                                                          06034
      adstr[40];                                                  06035
   LOCAL TEXT POINTER tpt1;                                       06036
   LOCAL STRING locstr[50];                                       06037
   REF stptr;                                                     06038
   % set up local string and text pointer%                        06039
      *locstr* _ "< , *stptr* , ",>";                             06040
      FIND SF(*locstr*) ^tpt1;                                    06041
      lnkprs($tpt1,$adstr);                                       06042
      RETURN(gpget($adstr, FALSE % don't print out messages %));  06043
   END.                                                           06044
(gtctrlbits) PROCEDURE  % return ctrl bits for new subsystem %    05987
   (dptptr, rule);                                                05988
   LOCAL instptr;                                                 05989
   REF dptptr, instptr, rule;                                     05990
   % check to make sure that we've been given a valid pointer %   05991
      IF dptptr.dptvalid # $dptvldationcode THEN                  05992
         RETURN( FALSE );                                         05993
   % set up to go through rule %                                  05994
      &instptr _ &rule;                                           05995
   %if there are no defined subsystems then return 7%             05996
      IF instptr.opcode # $xecute THEN RETURN( 7 );               05997
      &instptr _ instptr.addr;                                    05998
   %special check because "TENEX" is not in nlssubs%              05999
      IF *[dptptr.dptname]*[1] = 'T THEN RETURN(3);               06000
   % loop through rule %                                          06001
      WHILE &instptr DO                                           06002
         IF instptr.opcode = $keyop THEN                          06003
            IF *[instptr.addr]* = *[dptptr.dptname]* THEN         06004
               RETURN( instptr.ctrl )  % replacing existing subsys %
                                                                  06005
```

```
         ELSE IF *[instptr.addr]*[1] = *[dptptr.dptname]*[1] THEN
                                                                    06006
            % first char conflict %                                 06007
               IF instptr.ctrl.llcmd THEN RETURN( 3 )               06008
               ELSE &instptr _ instptr.alternative                  06009
      ELSE &instptr _ instptr.alternative;                          06010
% not found so return all bits on %                                 06011
   RETURN( 7 );                                                     06012
END.
                                                                    06013
(gpawdsleft) PROCEDURE %try to adjust buffer size so there are
wordsneeded words left%                                             06014
   (wordsneeded, %number of unused words to leave%                  06017
   downf); %adj. downward also%                                     06018
   LOCAL                                                            06016
      wordsleft,  %in user program buffer%                          05443
      wordstoadd,                                                   05445
      pagestoadd,                                                   05446
      remainder,                                                    05447
      successf,                                                     06022
      newbsize;                                                     05448
   % Compute space left in user buffer %                            05449
      wordsleft _ upgbend - upgffbuf + 1;                           05450
   %adjust buffer size so it will fit (hopefully)%                  05454
      IF wordsneeded < wordsleft AND NOT downf THEN RETURN(TRUE);   05455
      wordstoadd _ wordsneeded - wordsleft;                         05456
      pagestoadd _ (wordstoadd + 511) / 512;                        05457
      newbsize _ upgbsz + pagestoadd;                               05458
      successf _ TRUE;                                              06028
      IF newbsize > maxbsize THEN                                   06023
         BEGIN                                                      06024
         newbsize _ maxbsize;                                       06026
         successf _ FALSE;                                          06027
         END;                                                       06025
      gpbsz(newbsize);                                              05459
      RETURN(successf);                                             05460
   END.
                                                                    06015
(gpadjbsz) PROCEDURE % adjusts user program buffer size %           05439
   (jfn); %jfn of file which is going to be loaded%                 05440
   LOCAL                                                            05441
      wordsneeded,                                                  06019
      flength; %byte size of rel file, used as estimate of core size%
                                                                    05442
   % Estimate core size of the rel file%                            05451
      !gtfdb(jfn,1B6+$fdbsiz,$flength);                             05452
      wordsneeded _ flength + (flength/4)  %a 25 percent fudge factor%;
                                                                    05453
   gpawdsleft(wordsneeded, FALSE); %adjust it%                      06020
   RETURN (TRUE);                                                   06021
   END.                                                             05461
(gpl10)    PROCEDURE   % does Goto Program L10 program compile %    01967
   (stid, % stid where compilation is to start %                    01968
   da);   % display area descriptor %                               01969
                                                                    01970
   % Compile a L10 program.  If the compilation is successful and there
```

```
   is room in the buffer for the code and room in the user program
   stack, add the program to the stack of user programs, but do not make
   it the current CONAN program.  Mark DDT's symbol table. %        01971
                                                                    01972
   LOCAL size, errs;                                                01973
   LOCAL TEXT POINTER tp, tpb, tpe;                                 01974
   LOCAL STRING str[50];                                            01975
   REF da;                                                          01976
                                                                    01977
   IF NOT (FIND SF(stid) ^tp [("PROGRAM"/"FILE")] $NP ^tpb $LD ^tpe)
   THEN                                                             01978
      SIGNAL (upgerr, $"couldn't find program name");               01979
   *str* _ tpb tpe;                                                 01980
   astruc( $str );                                                  02122
   marksym ($str);                                                  01981
   [upgffbuf] _ upgbend - upgffbuf;   % space left in buffer %      01982
   errs _ <SEQFIL, processor>                                       01983
      (IF exp THEN $x110nam ELSE $110nam, $tp, &da, upgtyp, upgffbuf, 0
      :size);                                                       01984
   IF errs > 0 THEN                                                 01985
      BEGIN                                                         01986
      popsym();                                                     01987
      RETURN (errs)                                                 01988
      END                                                           01989
   ELSE IF upgskix >= $upgsksz THEN                                 01990
      BEGIN                                                         01991
      popsym();                                                     01992
      SIGNAL (upgerr, $"no more room in user program stack")        01993
      END                                                           01994
   ELSE                                                             01995
      % compilation was successful and there was room in the buffer and
      there is room in stack %                                      01996
      BEGIN                                                         01997
      upgstk [upgskix _ upgskix + 1] _ upgffbuf;                    01998
      upgffbuf _ upgffbuf + size;                                   01999
      *upgnms* _ *upgnms*, SP, *str*;                               02000
      END;                                                          02001
   RETURN (0);                                                      02002
   END.                                                             02003
                                                                    02004
(gppop)    PROCEDURE;  % does Goto Program Pop a program %         05675
   % Delete top program on user program stack --                     05676
      pop DDT's symbol table                                        05677
      find any references to it and delete them                     05678
      give back its space to the buffer                             05679
      fix the stack pointer and the string of program names %       05680
   LOCAL oldend, i, value, bpa;                                     05681
   LOCAL STRING pname[40];                                          05682
   LOCAL TEXT POINTER tp, tp1, tp2;                                 05683
   IF upgskix <= 0 THEN SIGNAL (upgerr, $"stack is empty");         05684
   FIND SE(*upgnms*) $NP ^tp2 $PT ^tp1 $NP ^tp;                     05685
   % get name of last program (not needed) %                        05686
      % *pname* _ tp1 tp2;) %                                       05687
   % detach the subsystem %                                         05688
      ON SIGNAL ELSE GOTO gpgt1;                                    05689
      IF (value _ upgstk[upgskix].LH := 0) THEN                     05690
```

```
      BEGIN                                                          05691
      delsubsys(value, $nlssubs);                                    05692
      delsubsys(value, $allsubs);                                    05693
      END;                                                           05694
   (gpgt1): ON SIGNAL ELSE;                                          05695
popsym ();                                                          05696
dinstupg (upgstk[upgskix]);                                         05697
IF upgstk[upgskix] IN [upgbuf, upgbend) THEN                        05698
   % if program is in buffer, free the buffer space %               05699
   oldend _ upgffbuf := upgstk[upgskix];                            05700
BUMP DOWN upgskix;                                                  05701
% clear any breakpoints here %                                      05702
   FOR i _ 0 UP UNTIL >= 10 DO                                      05703
      IF bpa _ findbp(i) THEN                                       05704
         IF ([[bpa]].LH = 265B3 % JSP %) AND                        05705
            ([bpa].RH IN [upgffbuf, oldend]) THEN                   05706
               BEGIN                                                05707
               [[bpa]] _ [bpa+1];                                   05708
               [bpa] _ [bpa+1] _ 0;                                 05709
               END;                                                 05710
% do procedure backups for any procedures in this program %        05711
   FOR i _ 0 UP 2 UNTIL >= 10 DO                                    05712
      IF proctbl[i] THEN                                            05713
         IF ([proctbl[i]].LH = 254B3) AND                           05714
            ([proctbl[i]].RH IN [upgffbuf, oldend]) THEN            05715
               BEGIN                                                05716
               [proctbl[i]] _ proctbl[i+1];                         05717
               proctbl[i] _ 0;                                      05718
               END;                                                 05719
IF upgstk[upgskix].LH  =18M THEN %was the program now on the top of
the stack loaded as a subsys which automatically pulled in a cml over
it? if so we should delete the subsys now because we just deleted the
cml.%                                                               05720
   BEGIN                                                            05721
   upgstk[upgskix].LH _ 0;                                          05722
   gppop();                                                         05723
   END                                                              05724
ELSE *upgnms* _ SF(*upgnms*) tp;                                    05725
RETURN;                                                             05726
END.                                                                05727
                                                                    05728

(gpstatus) PROCEDURE   % does Goto Program Status display %        02024
   (str, da);   % address of string in which to put stuff %         02025
   %make up a string containing the status of the user program stuff%
                                                                    02026
   REF str, da;                                                     02027
   *str* _                                                          02028
      "Stack of compiled programs (first is #1):", CR, LF,          02029
         "   ", *upgnms*, CR, LF, CR, LF;                           02030
   *str* _ *str*,                                                   02031
      "Content Analyzer  ", " program for display area:  ";         02032
   upgsstr (da.dacacode, &str);                                     02033
   *str* _ *str*,                                                   02034
      "Sequence Generator", " program for display area:  ";         02035
   upgsstr (da.dausqcod, &str);                                     02036
   *str* _ *str*,                                                   02037
```

```
   "Sort Key Extractor", " program for display area:  ";                02038
upgsstr (da.daukeycod, &str);                                           02039
*str* _                                                                 02040
   *str*, CR, LF, CR, LF,                                               02041
   "Current buffer size: ", STRING(upgbsz), " pages = ", STRING
   (512*upgbsz), " words.  ", CR, LF,                                   02042
   "Room left in buffer:  ",                                            02043
      STRING (upgbend - upgffbuf + 1),  " words.", CR, LF;              02044
RETURN;                                                                 02045
END.                                                                    02046
                                                                        02047
(upgsstr) PROCEDURE (pgmaddr, str);                                     06045
   LOCAL i,j;                                                           06046
   LOCAL TEXT POINTER tp1, tp2;                                         06047
   LOCAL STRING lstr[50];                                               06048
   REF str;                                                             06049
   IF pgmaddr = 0 THEN *lstr* _ "None"                                  06050
   ELSE                                                                 06051
      BEGIN                                                             06052
      *lstr* _ NULL;                                                    06053
      j _ 0;                                                            06054
      FOR i _ 1 UP UNTIL > upgskix DO                                   06055
         BEGIN                                                          06056
         IF upgstk[i] = pgmaddr THEN                                    06057
            BEGIN                                                       06058
            i _ i - j;  %subtract # of subsystems%                      06059
            FIND SF(*upgnms*);                                          06060
            FOR i _ i - 1 DOWN UNTIL = 0 DO FIND $NP $PT;               06061
            FIND $NP ^tp1 $PT ^tp2;                                     06062
            *lstr* _ tp1 tp2;                                           06063
            EXIT LOOP 1;                                                06064
            END;                                                        06065
         IF upgstk[i].LH = 777777B THEN BUMP j;  %subsystem (two
         words)%                                                        06066
         END;                                                           06067
      IF lstr.L = 0 THEN                                                06068
         IF NOT ddsymget ($lstr, pgmaddr) THEN *lstr* _ "None";         06069
      END;                                                              06070
   *str* _ *str*, *lstr*, CR, LF;                                       06071
   RETURN;                                                              06072
   END.                                                                 06073
                                                                        06074
(gptxst)          % does: SHOW TENEX SUBSYTEM STATUS %                  02332
   PROCEDURE                                                            02333
      ( fork,     % fork handle %                                        02334
      astr        % address of string to receive status message %       02335
      );                                                                02336
   LOCAL                                                                02337
      status,     % inferior fork's status %                             02338
      pc;         % inferior fork's pc %                                 02339
   REF astr;                                                            02340
                                                                        02341
   % RETURNS %                                                         02342
      % returns FALSE for bad fork handle input %                       02343
                                                                        02344
   IF NOT fork THEN                                                     02345
```

```
   BEGIN                                                                02346
   *astr* _ "Fork Does Not Exist";                                      02347
   RETURN( FALSE );                                                     02348
   END;                                                                 02349
!rfsts( fork );                                                         02350
IF r1.LH = 777777B THEN                                                 02351
   BEGIN                                                                02352
   *astr* _ "Fork Does Not Exist";                                      02353
   RETURN( FALSE );                                                     02354
   END;                                                                 02355
status _ r1.LH .A 377777B;                                              02356
pc _ r2.RH;                                                             02357
IF r1 .A 4B11 THEN *astr* _ "Interrupted from ";                        02358
CASE status OF                                                          02359
   = 0: *astr* _ *astr*, "Running";                                     02360
   = 1: *astr* _ *astr*, "IO Wait";                                     02361
   = 2: *astr* _ *astr*, "Halted";                                      02362
   = 3:                                                                 02363
      BEGIN                                                             02364
      *astr* _ *astr*, "Halted Because ";                               02365
      CASE r1.RH OF                                                     02366
         IN [0, 5], IN [12, 14], IN [21, 35], = 8:                      02367
            *astr* _ *astr*, "Channel ", STRING( r1.RH), "
            Interrupt";                                                 02368
         = 6:                                                           02369
            *astr* _ *astr*, "Overflow";                                02370
         = 7:                                                           02371
            *astr* _ *astr*, "Floating Overflow";                       02372
         = 9:                                                           02373
            *astr* _ *astr*, "Pushdown Overflow";                       02374
         = 10:                                                          02375
            *astr* _ *astr*, "End of File";                             02376
         = 11:                                                          02377
            *astr* _ *astr*, "IO Data Error";                           02378
         = 15:                                                          02379
            *astr* _ *astr*, "Illegal Instruction";                     02380
         = 16:                                                          02381
            *astr* _ *astr*, "Illegal Memory Read";                     02382
         = 17:                                                          02383
            *astr* _ *astr*, "Illegal Memory Write";                    02384
         = 18:                                                          02385
            *astr* _ *astr*, "Illegal Memory Execute";                  02386
         = 19:                                                          02387
            *astr* _ *astr*, "Fork Termination Interrupt";              02388
         = 20:                                                          02389
            *astr* _ *astr*, "Disk Space Allocation Exceeded";          02390
         ENDCASE;                                                       02391
      END;                                                              02392
   = 4: *astr* _ *astr*, "Fork Wait";                                   02393
   = 5: *astr* _ *astr*, "Sleep";                                       02394
   = 6: *astr* _ *astr*, "Breakpoint";                                  02395
   = 100: *astr* _ *astr*, "Processing Suspended";                      02396
   ENDCASE;                                                             02397
*astr* _ *astr*, " at ", STRING(pc, 8);                                 02398
RETURN( TRUE );                                                         02399
END.
```

```
                                                                    02400
(gpkill)        % does: KILL TENEX SUBSYSTEM %                      04400
   PROCEDURE;                                                       04401
   % deactivate inferior termination psi %                          04402
      IF chntab[19] = $fkterm THEN                                  04403
         BEGIN                                                      04404
         !dic( 400000B, 285);                                       04405
         chntab[19] _ 0;                                            04406
         END;                                                       04407
   % now get rid of the inferior %                                  04408
      IF infork THEN !kfork( infork := 0 );                         04409
   % close and release any lingering jfns %                        04410
      IF pjfn THEN       % subsystem %                             04411
         BEGIN                                                      04412
         IF NOT SKIP !closf( pjfn ) THEN NULL;                      04413
         IF NOT SKIP !rljfn( pjfn := 0 ) THEN NULL;                 04414
         END;                                                       04415
      IF ojfn > 0 THEN  % output file %                             04416
         BEGIN                                                      04417
         IF NOT SKIP !closf( ojfn ) THEN NULL;                      04418
         IF NOT SKIP !rljfn( ojfn := 0 ) THEN NULL;                 04419
         END;                                                       04420
      IF ijfn > 0 THEN  % input file %                              04421
         BEGIN                                                      04422
         IF NOT SKIP !closf( ijfn ) THEN NULL;                      04423
         IF NOT SKIP !rljfn( ijfn := 0 ) THEN NULL;                 04424
         END;                                                       04425
    % all done %                                                   04426
      RETURN;                                                       04427
    END.                                                            04428
                                                                    04429
    %  %
```

```
(gprunt)        % does: RUN TENEX SUBSYSTEM %                                  04430
                                                                               04431
   PROCEDURE
      (adstr,   % parsed link data structure for subsystem to be run %
                                                                               04432
                                                                               04433
      rhosto,   % host number for output file %                                04434
      outfil,   % null string for tty output; else file name %
      mode,     % 3 - file; 1 - interactive; 2 - typeahead; 4 - none %
                                                                               04435
      rhostt,   % host number for input or typeahead file %                    04436
      tahd,     % file name / typeahead / termination character %             04437
      wtmode    % TRUE for wait for completion %                              04438
                                                                               04439
      );                                                                       04440
   LOCAL
      intflg;   % flag indicating current status %                            04441
   LOCAL STRING                                                                04442
      tstring[40],      % temp string %                                        04443
      locstr[500];      % temp string %                                        04444
   REF subnam, outfil, tahd, adstr;                                            04445
                                                                               04446
   % can only run one inferior at a time %                                     04447
      IF infork THEN                                                           04448
         BEGIN                                                                 04449
         dismes( 2, $"Tenex Subsytem Already Running");                        04450
         RETURN;                                                               04451
         END;                                                                  04452
   % do some intializing and state saving %                                    04453
      !rfmod( 100B );                                                          04454
         tmode _ r2;                                                           04455
      !getnm();                                                                04456
         nlssbn _ r1;                                                          04457
      trmcod _ 0;                                                              04458
   % get a jfn for the subsystem first %                                       04459
      IF NOT ( pjfn _ upgjfn(&adstr) ) THEN                                    04460
         err($"Can't get jfn for this subsystem");                             04461
   % get jfn and open output file if asked for %                              04462
      IF NOT outfil.L THEN ojfn _ -1                                           04463
      ELSE                                                                     04464
         CASE rhosto OF                                                        04465
            = lhostn:                                                          04466
               BEGIN                                                           04467
               IF NOT (ojfn _ sgtjfn( gtjoof, &outfil, $lit)) THEN             04468
                  err($"Can't get jfn for output file");                       04469
               IF NOT sysopen( ojfn, append, chrtyp, $lit) THEN                04470
                  err($"Can't open output file");                              04471
               END;                                                            04472
            ENDCASE                                                            04473
               err($"Remote File Manipulations Not Implemented Yet");
                                                                               04474
                                                                               04475
    % get terminating character or %                                           04476
    % open input file or %                                                     04477
    % create temporary input file if asked for %                               04478
      CASE mode OF                                                             04479
         = 3:  % input from file mode %                                        04480
            CASE rhostt OF                                                     04481
               = lhostn:                                                       04482
                  BEGIN
```

```
            IF NOT (ijfn _ sgtjfn( gtjoif, &tahd, $lit)) THEN
                                                                04483
               err($"Can't get jfn for input file");            04484
            IF NOT sysopen( ijfn, read, chrtyp, $lit) THEN      04485
               err($"Can't open input file");                   04486
            END;                                                04487
         ENDCASE                                                04488
            err($"Remote File Manipulations Not Implemented Yet");
                                                                04489
   = 1:  % interactive mode %                                   04490
      BEGIN                                                     04491
      ijfn _ -1;                                                04492
      % check for valid termination character %                 04493
         CASE (intflg _ *tahd*[1]) OF                           04494
            IN [1,33]: NULL;       % ^A through ^Z %           04495
            = 40: intflg _ 29;     % space %                   04496
            = 177: intflg _ 28;    % rubout %                  04497
            = ENDCHR: intflg _ 0; % no termination char %      04498
            ENDCASE                                             04499
               err($"Illegal Temination Character Specified");
                                                                04500
      END;                                                      04501
   = 2, = 4:  % none or typeahead mode %                        04502
      BEGIN                                                     04503
      mode _ 2;   % look like typeahead from here on out %      04504
      IF NOT tahd.L THEN ijfn _ 377777    % nil input %         04505
      ELSE                                                      04506
         CASE rhostt OF                                         04507
           = lhostn:                                            04508
            BEGIN                                               04509
            % get string name for temporary file %              04510
               jfntostr( pjfn, $tstring, 001000B6);             04511
               *locstr* _ "<, *userstr*, ">, *tstring*,
               "-TYPEAHEAD.", *initsr*, ";T";                   04512
            IF NOT (ijfn _ sgtjfn( gtjoof, $locstr, $lit)) THEN
                                                                04513
               err($"Can't get jfn for temporary input file");
                                                                04514
            IF NOT sysopen( ijfn, readwrite, chrtyp, $lit) THEN
                                                                04515
               err($"Can't open temporary input file");         04516
            !sout( ijfn, chbmty+&tahd, -tahd.L);                04517
            IF NOT SKIP !sfptr( ijfn, 0 ) THEN                  04518
               err($"Can't set byte pointer for temporary input
               file");                                          04519
            END;                                                04520
         ENDCASE                                                04521
            err($"Remote File Manipulations Not Implemented
            Yet");                                              04522
      END;                                                      04523
   ENDCASE err($"Illegal Input Mode");                          04524
% create the inferior fork %                                   04525
   IF NOT SKIP !cfork( 2B11 ) THEN                              04526
      err($"Can't create inferior fork");                       04527
   infork _ r1;                                                 04528
% enable all capabilities %                                    04529
```

```
   !epcap( infork, -1, -1 );                                          04530
% get the subsystem (and get rid of jfn ) %                          04531
   r1.LH _ infork;                                                    04532
   r1.RH _ pjfn;                                                      04533
   !get( r1 );                                                        04534
   IF NOT SKIP !rljfn( pjfn ) THEN NULL;                              04535
% now start it; for interactive mode, shut down dnls first %        04536
   CASE mode OF                                                       04537
      = 3, =2:        % file or typeahead mode %                     04538
         BEGIN                                                        04539
         % setup primary jfns %                                      04540
            r2.LH _ ijfn;                                           04541
            r2.RH _ ojfn;                                           04542
            !spjfn( infork, r2);                                    04543
         % set up to receive fork termination psi %                  04544
            IF NOT wtmode THEN                                      04545
               BEGIN                                                04546
               chntab[19] _ 1B6 + $fkterm;                          04547
               !aic( 400000B, 200000);                              04548
               END;                                                 04549
         % disable all terminal psi for inferior %                  04550
            !stiw( infork, 0);                                      04551
         % start the fork %                                         04552
            !sfrkv( infork, 0);                                     04553
         % return (after waiting if appropriate %                   04554
            IF wtmode THEN                                          04555
               BEGIN                                                04556
               !wfork( infork );                                    04557
               gpkill();                                            04558
               !setnm( nlssbn );                                    04559
               !sfmod( 100B, tmode := 0 );                          04560
               END;                                                 04561
            RETURN;                                                 04562
         END;                                                       04563
      = 1:            % interactive mode %                          04564
         BEGIN                                                      04565
         % shut down dnls %                                         04566
            IF nlmode = fulldisplay THEN shutdis();                 04567
         % save terminal mode %                                     04568
            !rfmod( 100B );                                         04569
            tmode _ r2;                                             04570
         % reset terminal modes %                                   04571
            !sfmod( 100B, 20510175520B);                            04572
         % set up psi system %                                      04573
            IF (trmcod _ intflg) THEN                               04574
               BEGIN  % termination char specified %                04575
               % set up to receive fork termination psi %           04576
                  IF NOT wtmode THEN                                04577
                     BEGIN                                          04578
                     chntab[19] _ 1B6 + $fkterm;                    04579
                     !aic( 400000B, 200000);                        04580
                     END;                                           04581
               % save terminal interrupt word %                     04582
                  !rtiw( 400000B );                                 04583
                  tiw _ r2;                                         04584
               % setup termination char as psi on channel 35 %      04585
```

```
                chntab[35] _ 1B6 + $trmpsi;                          04586
                r1.LH _ trmcod;                                      04587
                r1.RH _ 35;                                          04588
                !ati( r1 );                                          04589
                !aic( 400000B, 1);                                   04590
             % disallow all psi chars (for us) except terminating
             character %                                             04591
                r1 _ 35 - trmcod;                                    04592
                r2 _ 1;                                              04593
                !LSH r2,0(r1);                                       04594
                !stiw( 400000B, r2);                                 04595
             % make sure we get the psi and not the inferior %
                                                                     04596
                !sircm( infork, 1);                                  04597
             END                                                     04598
          ELSE                                                       04599
             BEGIN  % no termination char specified %                04600
             % save terminal interrupt word %                        04601
                !rtiw( 400000B );                                    04602
                tiw _ r2;                                            04603
             % disallow all psi chars (for us) except ^C %           04604
                !stiw( 400000B, 0);                                  04605
             END;                                                    04606
       % now start the fork %                                       04607
          !sfrkv( infork, 0);                                        04608
       END;                                                          04609
                                                                     04610
    ENDCASE;                                                         04611
% now wait for terminating char psi or until the fork finishes %
                                                                     04612
   % (get here only for interactive mode) %                         04613
   IF wtmode OR NOT trmcod THEN                                      04614
      BEGIN  % no termination char specified %                       04615
      % wait for inferior to terminate %                            04616
         !wfork( infork );                                           04617
      % resume nls %                                                04618
         gprnls( tmode := 0 );                                       04619
      % get back our psi tiw %                                      04620
         !stiw( 400000B, tiw := 0 );                                 04621
      % give user termination status %                              04622
         gptxst( infork, $locstr);                                   04623
         dismes(2, $locstr );                                        04624
      % now cleanup %                                               04625
         gpkill();                                                   04626
      % and we are done! %                                          04627
         RETURN;                                                     04628
      END                                                            04629
   ELSE      % termination char specified %                          04630
      !wait();       % this does not return!!! %                    04631
                                                                     04632
% we get here only when we get the terminating char psi %           04633
   (trmpsi):                                                         04634
   % freeze the inferior to avoid some problems %                   04635
      !ffork( infork );                                              04636
   % deactivate and deassign this psi %                             04637
      !dti( trmcod := 0 );                                           04638
```

```
      !dic( 400000B, 1);                                              04639
      chntab[35] _ 0;                                                 04640
   % setup primary jfns %                                             04641
      !gpjfn( infork );                                               04642
      IF r2.LH = 777777B THEN r2.LH _ 377777B;                        04643
      r2.RH _ IF ojfn > 0 THEN ojfn ELSE 777777B;;                    04644
      !spjfn( infork, r2);                                            04645
   % disable all terminal psi for the inferior %                      04646
      !stiw( infork, 0);                                              04647
   % get back our proper tiw %                                        04648
      !stiw( 400000B, tiw := 0);                                      04649
   % now get back into NLS %                                          04650
      gprnls( tmode );                                                04651
   % setup to debrk to sysrtn %                                       04652
      IF NOT wtmode THEN [levtab] _ $sysrtn;                          04653
   % now resume the inferior %                                        04654
      !rfork( infork );                                               04655
   % now debrk %                                                      04656
      !debrk();                                                       04657
                                                                      04658
% get here on fork termination psi %                                 04659
   (fkterm):                                                          04660
   % if its the fork we are interested in notify the user and
   cleanup, else do nothing %                                         04661
      IF NOT infork THEN !debrk();                                    04662
      !rfsts( infork );                                               04663
      CASE r1.LH .A 377777B OF                                        04664
         = 2:                                                         04665
            BEGIN                                                     04666
            IF trmcod THEN                                            04667
               BEGIN  % we got fork term psi before term char psi %
                                                                      04668
               % deactivate and deassign term char psi %              04669
                  !dti( trmcod := 0 );                                04670
                  !dic( 400000B, 1);                                  04671
                  chntab[35] _ 0;                                     04672
               % get back our proper tiw %                           04673
                  !stiw( 400000B, tiw := 0);                          04674
               % now get back into NLS %                             04675
                  gprnls( tmode );                                    04676
               % get rid of the inferior %                           04677
                  gpkill();                                           04678
               % setup to debrk to sysrtn %                          04679
                  [levtab] _ $sysrtn;                                 04680
               END                                                    04681
            ELSE                                                      04682
               % get rid of the inferior %                           04683
                  gpkill();                                           04684
            dismes(2, $"Tenex Subsystem Completed");                  04685
            END;                                                      04686
         = 3:                                                         04687
            BEGIN                                                     04688
            IF trmcod THEN                                            04689
               BEGIN  % we got fork term psi before term char psi %
                                                                      04690
               % deactivate and deassign term char psi %              04691
```

```
                !dti( trmcod := 0 );                                    04692
                !dic( 400000B, 1);                                      04693
                chntab[35] _ 0;                                         04694
             % get back our proper tiw %                                04695
                !stiw( 400000B, tiw := 0);                              04696
             % now get back into NLS %                                  04697
                gprnls( tmode );                                        04698
             % tell the user what happened %                            04699
                gpfktrm();                                              04700
             % get rid of the inferior %                                04701
                gpkill();                                               04702
             % setup to debrk to sysrtn %                               04703
                [levtab] _ $sysrtn;                                     04704
             END                                                        04705
          ELSE                                                          04706
             BEGIN                                                      04707
             % tell the user what happened %                            04708
                gpfktrm();                                              04709
             % get rid of the inferior %                                04710
                gpkill();                                               04711
             END;                                                       04712
          END;                                                          04713
       ENDCASE;                                                         04714
    !debrk();                                                           04715
                                                                        04716
END.                                                                    04717
% %                                                                    04718
```

```
% support routines for doing procedure replace and backup %          03214
   (rpiproc) % Replace procedure oldname by newname %                03855
      PROCEDURE                                                      03856
         (oldname, newname);                                         03857
      LOCAL oldval, newval, c;                                       03858
      REF oldname, newname;                                          03859
                                                                     03860
      c _ 0;                                                         03861
      IF NOT (oldval _ jutval(&oldname, TRUE : c)) THEN              03862
         err($"No old procedure: error in DDT lookup.");             03863
      IF NOT (newval _ jutval(&newname, FALSE)) THEN                 03864
         err($"No new procedure: error in DDT lookup.");             03865
      IF oldval = newval THEN RETURN( FALSE );                       03866
      % Put "old" procedure in backup table. %                       03867
         IF NOT c THEN                                               03868
            IF NOT tblsearch($proctbl, 30 %proctblsz%, 2 %proctblentsz%,
            0: c) THEN                                               03869
               err($"Backup Table Full...Nothing Replaced");         03870
         [c] _ oldval;                                               03871
         [ c +1] _ [ oldval];                                        03872
         [oldval] _ 254B9 %jrst% + newval;                           03873
      RETURN( TRUE );                                                03874
      END.
                                                                     03875
   (jutval)     % find value of name %                               03235
      PROCEDURE                                                      03236
         (name, mode);                                               03237
      LOCAL retval, c;                                               03238
      REF name;                                                      03239
                                                                     03240
      IF NOT ddtlookup(&name, mode: retval) THEN                     03241
         RETURN(FALSE);                                              03242
      IF ([retval].accum10 # $s AND [retval].addr10 # $sysovr) THEN
                                                                     03243
         IF [retval].opcod10 # 254B %JRST% THEN RETURN(FALSE)        03244
         ELSE % Check if breakpoint or procedure replace done already %
                                                                     03245
            IF tblsearch($proctbl, 30 %proctblsz%, 2 %proctblentsz%,
            retval:c) THEN RETURN(retval, c)                         03246
            ELSE RETURN(FALSE);                                      03247
      RETURN(retval, 0);                                             03248
      END.
                                                                     03249
   (bkproc)     % do procedure backup %                              03250
      PROCEDURE( name );                                             03260
      LOCAL oldval, c;                                               03251
      REF name;                                                      03252
                                                                     03253
      IF NOT (oldval _ jutval(&name, TRUE :c)) THEN RETURN(FALSE);   03254
      IF c = 0 THEN RETURN(FALSE);                                   03255
      [oldval] _ [c+1]; % Original backup instruction. %             03256
      [c] _ 0; % Clear it in table. %                                03257
      RETURN(TRUE);                                                  03258
      END.
                                                                     03259
% support routines for running inferior tenex subsystems %          02716
```

```
(gptktrm)    % print tenex subsytem status from within fork
termination psi routine %                                                02717
   PROCEDURE;                                                            02718
   LOCAL STRING                                                          02719
      tstring[500],                                                      02720
      strng[500];                                                        02721
                                                                         02722
   gptxst( infork, $strng);                                              02723
   *tstrng* _ CR, LF, "Tenex Subsystem Aborted", CR, LF, *strng*;
                                                                         02724
   dismes( 2, $tstrng);                                                  02725
   RETURN;                                                               02726
                                                                         02727
   END.
                                                                         02728
   %  %                                                                  02729
```

```
(gprnls)     % restore nls display %                                    02730
   PROCEDURE                                                           02731
      ( ttymode      % teminal mode to be set up %                     02732
      );                                                               02733
                                                                       02734
   % restore subsystem name %                                          02735
      !setnm( nlssbn );                                                02736
   % restore terminal mode words %                                     02737
      !sfmod( 100B, ttymode);                                          02738
   % restore the display %                                             02739
      continue _ TRUE;                                                 02740
      initdis();                                                       02741
      continue _ FALSE;                                                02742
      dpset( dspallf, endfil, endfil, endfil);                         02743
   % should be done, so return %                                       02744
      RETURN;                                                          02745
                                                                       02746
   END.                                                                02747

   %  %                                                               02748
```

```
(upgcnv)   PROCEDURE   % get user program address from name or number%
                                                                    05731
   (str);   % address of string containing name or number %         05732
   % given the name or number of a user program in a string, look up the
   name in the string of user program names and/or the address in the
   stack of user program starting addresses %                       05733
   LOCAL pgmidx, i ,j;                                              05734
   REF str;                                                         05735
   IF str.L = empty THEN pgmidx _ 0                                 05736
   ELSE IF *str* [1] IN ['0, '9] THEN                               05737
      BEGIN                                                         05738
      pgmidx _ cvsno (&str);                                        05739
      % check to see if there are any user subsytems loaded, and adjust
      the pgmidx accordingly. User subsystems appear as one program to
      the user but actually take up two entries in the upgstk stack%
                                                                    05740
      j _ 1;                                                        05741
      FOR i_1 UP 1 UNTIL  >= pgmidx DO                              05742
         BEGIN                                                      05743
         IF upgstk[j].LH = 777777B THEN BUMP j;                     05744
         BUMP j;                                                    05745
         END;                                                       05746
      pgmidx _ j;                                                   05747
      END                                                           05748
   ELSE                                                             05749
      BEGIN                                                         05750
      astruc( &str );                                               05751
      pgmidx _ upgskix;                                             05752
      FIND SE(*upgnms*) $NP;                                        05753
      WHILE NOT (FIND < ENDCHR) DO                                  05754
         BEGIN                                                      05755
         FIND < $NP $PT;   % move to next visible %                 05756
         IF (FIND > *str* (NP/ENDCHR)) THEN EXIT LOOP               05757
         ELSE BUMP DOWN pgmidx;                                     05758
         IF pgmidx > 1 AND upgstk[ pgmidx-1].LH = 777777B THEN BUMP DOWN
         pgmidx;                                                    05759
         END;                                                       05760
      END;                                                          05761
   IF NOT pgmidx IN [1, upgskix] THEN                               05762
      SIGNAL (upgerr, $"illegal user program spec");                05763
   RETURN (upgstk[pgmidx]);                                         05764
   END.                                                             05765
                                                                    05766
(popsym)   PROCEDURE;  % pops block of symbols from DDT's symbol table %
                                                                    02094
   ddtpop();                                                        02095
   RETURN;                                                          02096
   END.
                                                                    02097
(marksym)  PROCEDURE   % marks DDT's symbol table %                 02098
   (blocknm);   % 0 or string containing the block name %
                                                                    02099
   LOCAL STRING str[50];                                            02100
   REF blocknm;                                                     02101
   IF &blocknm THEN *str* _ *blocknm*                               02102
   ELSE *str* _ "LOCAL";                                            02103
```

```
   astruc( $str ); % force to upper case %                              02104
   ddtmark($str);                                                       02105
   RETURN;                                                              02106
   END.
                                                                        02107
(upgjfn)        % get a jfn for parsed link data structure %            02912
   PROCEDURE                                                            02914
      (adstr);  % address of parsed link data structure %               02913
   LOCAL try, jfn;                                                      03003
   LOCAL TEXT POINTER tp1, tp2;                                         02930
   LOCAL STRING filename[200], dirname[100];                            02916
   REF adstr;                                                           02917
                                                                        02918
   % ALGORITHM %                                                       02919
      % if a directory name is specified for the link then this
      directory is searched for the file.  if no directory name is
      specified then the following directories are searched for the file
      (in the following order):                                         02920
         NETSYS                                                         02921
         SUBSYS                                                         02922
         the default directory for links for the link                   02923
         the connected directory                                        02924
         the login directory                                            02925
      in all cases the default extension name is "SAV". %               02927
   % RETURNS %                                                         02928
      % this routine will return the jfn for the found file or FALSE;
      it will generate an err for links which specify a remote hostname
      %                                                                 02929
                                                                        02931
   % get file name as specified in link (& check for local host) %     02939
      CASE lnbfls( 0, &adstr, $filename) OF                             02947
         = lhostn: NULL;                                                02948
         ENDCASE err($"Remote File Manipulations Not Implemented Yet");
                                                                        02949
   % only try once if directory specified in link %                    02950
      IF adstr[ue+1] > adstr[us+1] THEN                                 02951
         BEGIN                                                          06084
         IF NOT (jfn _ lgetjfn(0, $filename, $savext, gtjprf, $lit) )
         THEN                                                           02952
            IF NOT (jfn _ lgetjfn(0, $filename, $exeext, gtjprf, $lit) )
            THEN                                                        06080
               err($lit);                                               06081
         RETURN(jfn);                                                   06082
         END;                                                           06083
   % now try various directories %                                     02954
      tp1 _ adstr[fs];  tp1[1] _ adstr[fs+1];                           02955
      tp2 _ adstr[fe];  tp2[1] _ adstr[fe+1];                           02956
      *filename* _ tp1 tp2;                                             02957
      try _ 0;                                                          02958
      *dirname* _ "NETSYS";                                             02959
      LOOP                                                              02970
         CASE (jfn_ lgetjfn($dirname,$filename,$savext,gtjprf,$lit)) OF
                                                                        02960
            = FALSE:                                                    02961
               CASE (jfn_
               lgetjfn($dirname,$filename,$exeext,gtjprf,$lit)) OF       06077
```

```
            = FALSE:                                                    06078
               CASE try _ try + 1 OF                                    02964
                  = 1:    % SUBSYS or NETSYS or whatever %              02965
                    *dirname* _ *subdir*;                               02971
                  = 2:    % SUBSYS %                                    06075
                    *dirname* _ *subdr2*;                               06076
                  = 3:    % default directory for links %               02966
                    IF NOT adstr[lfn] OR NOT gdftdir( adstr[lfn],
                    $dirname) THEN                                      02968
                    ELSE REPEAT CASE;                                   02969
                  = 4:    % connected directory %                       02973
                    *dirname* _ NULL;                                   02974
                  = 5:    % login directory %                           02975
                    *dirname* _ *userstr*;                              02976
                  ENDCASE err($lit);                                    02977
            ENDCASE RETURN( jfn );                                      06079
         ENDCASE RETURN( jfn );                                         02962
END.
                                                                        02963
FINISH                                                                  02108
```

CSENDMAIL

```
< NLS, CSENDMAIL.NLS;14, >, 1-AUG-77 15:43 JCP ;;;;( 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
(setjinfo) %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
                                                                       0102
(setjcomment) %set Comment field in JWORK header%
PROCEDURE (commentstr);                                                0103
   REF commentstr; %address of string containing desire comment%       0104
   LOCAL TEXT POINTER z1,z2;                                           0105
   %---------------%                                                  0106
   %remove " from the string%                                          0107
      FIND SF(*commentstr*);                                           0108
      LOOP CASE READC OF                                               0109
         = '": %replace it with a single quote%                        0110
            BEGIN                                                      0111
            FIND ^z2 < CH > ^z1;                                       0112
            ST z1 z2 _ '';                                             0113
            FIND z2 >;                                                 0114
            END;                                                       0115
         = ENDCHR: EXIT LOOP;                                          0116
         ENDCASE ;
   FIND jwp1 > ["Clerk: "] ["
                                                                       0117
   "] ^z1;
   ST jwp1 _ SF(jwp1) z1, ".PEL;  .PN=PN-1; .GCR;", *commentstr*;
                                                                       0118
   RETURN END.                                                         0119
                                                                       0120
(setjexpedite) %set or clear expedite attribute in JWORK header%
PROCEDURE (onoff);                                                     0121
   %IF onoff = TRUE, set expedite attribute, otherwise clear it%       0122
   LOCAL TEXT POINTER z1,z2;
```

```
   %---------------%                                                    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
      =textv:  %text%                                                    0193
         BEGIN                                                           0194
          % insert new source %                                          0195
            cinssta(jwp1, levdown, &s1, &s2);                            0196
         ST t1 t2 _ 'F;                                                  0197
         END;                                                            0198
      =stmtv:  %statement%                                               01529
         BEGIN                                                           01530
          % insert new source %                                          01531
            ccopsta(jwp1, levdown, s1, 0, 0);                            01532
         ST t1 t2 _ 'F;                                                  01533
         END;                                                            01534
      =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
                                                                        0259
   LOCAL TEXT POINTER z1, z2;
   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: "] /
      ["Distribution: "] / ["Author(s): "]) [';] ^z1 ^z2);              0264
   ST z1 z2 _ " Obsoletes Document(s): ", *obsoletestr*, '; ';          0265
   RETURN;                                                              0266
   END.
                                                                        0267
(setjunrecorded) %set or clear UNRECORDED attribute in JWORK heade %
                                                                        0268
PROCEDURE (onoff);
   %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
(setjupdates) %set Updates Document(s) field in JWORK header%
                                                                        0278
PROCEDURE (updatestr);                                                  0279
   LOCAL TEXT POINTER z1, z2;
   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
(setjtitle)%set Title field in JWORK header%
                                                                        0288
PROCEDURE(titlestr);                                                    0289
   REF titlestr; %address of string containing desired title%           0290
   LOCAL TEXT POINTER z1,z2;                                            0291
   %---------------%                                                   0292
   %remove " from the string%                                           0293
      FIND SF(*titlestr*);                                              0294
      LOOP CASE READC OF                                                0295
         = '": %replace it with a single quote%                         0296
            BEGIN                                                       0297
            FIND ^z2 < CH > ^z1;                                        0298
            ST z1 z2 _ '';                                              0299
            FIND z2 >;
```

```
            END;                                                    0300
         = ENDCHR: EXIT 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: .H1=", '"', *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 jwp1 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 CRLF'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
   *[addcrlf ( getjnumber ( $str ), $sjnumber )]*,                   0344
   *[addcrlf ( getjaxcess ( $str ), $sjaccess )]*,                   0345
   *[addcrlf ( getjtitle ( $str ), $sjtitle )]*,                     0346
   *[addcrlf ( getjcomment ( $str ), $sjcomment )]*,                 0347
   *[addcrlf ( getjauthor ( $str ), $sjauthor )]*,                   0348
   *[addcrlf ( getjaction ( $str ), $sjaction )]*,                   0349
   *[addcrlf ( getjinfo ( $str ), $sjinfo )]*,                       0350
   *[addcrlf ( getjsubcol ( $str,0 ), $sjsubcol  )]*;                0351
*status* _ *status*, %this is broken to avoid an L10 STACK
OVERFLOW%                                                            0352
   *[addcrlf ( getjkeyw ( $str ), $sjkeywords )]*,                   0353
   *[addcrlf ( getjexpedite ($str), $sjhandling)]*,                  0354
   *[addcrlf ( getjunrecorded ($str), $sjrecording)]*,               0355
   *[addcrlf ( getjrfc ( $str ), $sjrfc )]*,                         0357
   *[addcrlf ( getjobsoletes ( $str ), $sjobsoletes )]*,             0358
   *[addcrlf ( getjupdates ( $str ), $sjupdates )]*,                 0359
   *[addcrlf ( 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(jwp1) ["Clerk: "] ["Origin: "] ['<] ^z1 ['>] < CH
      > ^z2 THEN                                                     01259
         *status* _ *status*, CR, LF, *sjfile*, z1 z2, CR, LF
                                                                     01262
      ELSE  %see if it is a structure%                               01263
         IF (stid _ getnxt(jwp1)) NOT= endfil THEN % There is
         substructure%                                               01254
            IF getnxt(stid) = endfil THEN                            01255
               IF NOT FIND SF(stid) *nullsource* ENDCHR THEN         01264
                  *status* _ *status*, CR, LF, *sjmessage*,
                  SF(stid) SE(stid), CR, LF                          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*, CR, LF, "BRANCH AT: ", '",
                  z1 z2, '", CR, LF                                  01270
               ELSE                                                  01269
                  BEGIN                                              01275
                  *status* _ *status*, CR, LF, "GROUP AT: ", '",
                  z1 z2, '", CR, LF;                                 01274
                  stid _ getail(stid);                               01278
                  FIND SF(stid) ^z1 $15CH ^z2;                       01276
                  *status* _ *status*, "  [THROUGH]  ", '", z1 z2,
                  '", CR, LF;                                        01279
                  END;                                               01277
               END;                                                  01273
      END;                                                           01258
   = 'H:  %hardcopy%                                                 01284
      *status* _ *status*,                                           01286
         *[addcrlf ( getjhcloc ( $str ), $sjhardcopy )]*;            0356
```

```
      = 'S: %secondary distribution%                                    01285
        IF (stid _ getnxt(jwp1)) NOT= endfil THEN % There is
        substructure%                                                   01288
           *status* _ *status*, CR, LF, *sjforward*, SF(stid)
           SE(stid), CR, LF;                                            01287
      ENDCASE;                                                          01283
   RETURN (&status);                                                     0364
   END.
                                                                         0365
(addcrlf) %support routine for jstatus%
PROCEDURE (string, heading);                                             0366
   %*addeol*                                                             0367
      This routine is used by jstatus for inserting CRLF'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 a CRLF.              0369
      %                                                                  0370
   REF string, heading;                                                  0371
   %-----------------%                                                   0372
   IF string.L > 0 THEN *string* _ *heading*, SP, *string*, CR, LF;
                                                                         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
It 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
(getjlink) %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
(getjkeyw) %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
(getjnumber) %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
(getjtype) %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
(getjobsoletes) %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 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], temp1sr[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
               *temp1sr* _ +z2 z1;                                    01230
               IF NOT FIND SF(*string*) [*temp1sr*] THEN              01231
                  *string* _ *string*, SP, *temp1sr*;                 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, $z3);                                       0513
      IF NOT fndungtxt ($"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
        < 28000: catalog _ $"Jcat4";                                  01525
        < 32000: catalog _ $"Jcat5";                                  01526
        < 36000: catalog _ $"Jcat6";                                  01527
        < 40000: catalog _ $"Jcat7";                                  01528
        < 44000: catalog _ $"Jcat8";                                  01535
        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, distd1)%                             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");                           0730
      RETURN(flagut(0, $tstfg));                                       0731
      END.                                                             0732

   (lockjo)PROC(type);                                                01502
      %Type = 1 for bad file lock (flag 1), and 0 for jlock (flag 0);
                                                                       01503
      %                                                                01504
      IF type > 1 THEN err($"Illegal Flag Number");                    01505
      flagut(type, $setfg);                                            01506
      specttyout(0, $"Journal Locked");                                01507
       IF type = 1 THEN specttyout(oprtty, $"Journal Locked");         01508
      RETURN;                                                          01509
      END.                                                             01510
   (unlkjo)PROC(type);                                                 0742
      %Type indates flag to be unlocked... -1 means all%               0743
      IF type NOT IN [-1, 1] THEN err($"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 [10000], %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
IF nojournalflag THEN err($"Journal Not Enabled on this System");
                                                                       01501
jolock(); %Check to make sure it will work -- does not return
unless things are cool%                                                 0768
%set stid's to null%                                                    0769
   rnumstid _ numstid _ orgstid;                                        0770
capsav _ 0;                                                             0771
traping _ connflag _ FALSE;                                             0772
&orgstr _ 0;                                                           01289
ON SIGNAL                                                               0773
   ELSE                                                                 0774
      BEGIN %close any open files%                                      0775
      ON SIGNAL ELSE;                                                   0776
      conjdir(FALSE);                                                   0777
      IF numstid.stfile THEN close(numstid.stfile := 0);                0778
      IF rnumstid.stfile THEN close(rnumstid.stfile := 0);              0779
      IF traping := FALSE THEN notrapcc(capsav);                        0781
      IF connflag := FALSE THEN %connect back to user's directory%
                                                                        0782
         condir($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 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
%till in time and date (of submission) fields%                      0852
   *datesr* _ NULL;                                                 0853
   dtfrmt(-1, $datesr);                                             0855
   IF NOT FIND jwp1 > [";;;;"] [ "(J" / "(j" ] [';] ^p1 _p1
   ["Author(s):"] ['/] ^p3 [';] ^p4 _p4 THEN                        0856
      werr($"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) = $psprivate THEN                      0705
      BEGIN                                                        01195
      accesslist.L _ 0;                                             0707
      clnidlst (getjaction ($workstr), $accesslist);                0708
      clnidlst (getjinfo ($workstr), $accesslist);                  0709
      clnidlst (getjauthor ($workstr), $accesslist);                0710
      clnidlst (getjclerk ($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
                                                                             0878
      *dirnam* _ NULL;                                                       0879
      &fl _ flntadr(jworkstid.stfile);                                       0880
      jfnstr(fl.florig, $fnamestr);                                          0881
      traping _ TRUE;                                                        0882
      capsav _ trapcc();                                                     0883
      connflag _ TRUE;
      condir(IF jdebug THEN $"IRBV" 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, $fnamestr);                              0900
         resetf(jworkstid.stfile);                                           0901
         (jsubok): %initialize the file%                                     0902
         &fl _ flntadr(jworkstid.stfile);                                    0903
         fl.flnoclos _ TRUE;                                                 0904
         initjwork(); %zap the Journal workfile%                             0905
         FIND SF(jworkstid) ^jwp1;                                           0906
                                                                             0909
   ON SIGNAL                                                                 0910
      ELSE                                                                   0911
         BEGIN %close any open files%                                        0912
         ON SIGNAL ELSE;                                                     01306
         conjdir(FALSE);                                                     0913
         IF numstid.stfile THEN close(numstid.stfile := 0);                  0914
         IF rnumstid.stfile THEN close(rnumstid.stfile := 0);                0915
         IF idfno THEN close(idfno := 0);                                    0916
         IF traping := FALSE THEN notrapcc(capsav);                          0917
         END;                                                                0907
   IF number.L AND *number* # "1234" THEN                                    0908
      BEGIN                                                                  0918
      conjdir(TRUE);                                                         0919
      IF rfcnum.L THEN                                                       0920
         BEGIN %release RFC number%                                          0921
         usedcnum($rfcnum, rnumstid);                                        0922
         END;                                                                0923
      usedcnum($number, numstid); %release used journal number%              0924
      conjdir(FALSE);                                                        0925
      END;                                                                   0926
   IF linkaddr.L THEN                                                        0927
      BEGIN                                                                  0928
      CASE *itemtype*[1] OF
```

```
        = '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
   %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 _ flntadr(jworkstid.stfile);                                01447
         fl.flnoclos _ TRUE;                                             01448
         initjwork();  %zap the Journal workfile%                        01449
         FIND SF(jworkstid) ^jwp1;                                       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
         cdelgro (tailstrt, tailend, FALSE, 0);                     01084
      IF boundstid THEN                                             01085
         cdelgro (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, $nullsource, $"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
   (nioco) %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);                                 01511
      %Type message to Journal Logging file.                            01512
         Number = Number                                                01513
         typest is a string which describes use of number               01514
         identsr is the ident of author, etc.                           01515
      %                                                                 01516
      REF number, typest, identsr;                                      01517
      LOCAL STRING tempsr[200];                                         01518
      *tempsr* _ *number*, SP, *typest*, SP, *identsr*, SP;             01519
      dtfrmt(-1, $tempsr); %Add the date/time%                          01521
      specttyout(0, $tempsr);                                           01522
```

```
    RETURN;                                                    01523
    END.                                                       01524
FINISH      .                                                  01194
```

CSYNGEN

```
< NLS, CSYNGEN.NLS;3, >, 9-NOV-76 15:52 JDH ;;;;
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
         gtruln( 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, savstr;                                             01046
   LOCAL STRING locstr[200];                                           01047
   REF astr, anode, savstr;                                            01048
                                                                       01049
   &savstr _ 0;                                                        01113
   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
            IF astr.L <= locstr.M THEN                                 01099
               *locstr* _ *astr*                                       01100
            ELSE                                                       01101
               BEGIN                                                   01102
               &savstr _ getstring(astr.L, $dspblk);                   01103
               *savstr* _ *astr*;                                      01104
               END;                                                    01105
            goingdown _ nprint( &astr, anode.alternative, TRUE, altsuc);
                                                                       01057
            IF NOT &savstr THEN                                        01106
               *astr* _ *locstr*                                       01107
            ELSE                                                       01108
               BEGIN                                                   01109
               *astr* _ *savstr*;                                      01110
               freestring(&savstr := 0, $dspblk);                      01111
               END;                                                    01112
            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, savstr;                                  0822
   LOCAL STRING locstr[200];                                           0823
   REF astr, anode, savstr;                                            0824
                                                                       0825
   &savstr _ 0;                                                       01114
   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
= $xecute:        % 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
                                                                 0952
   lefthalf of word = -1
      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
                                                                 0954
   lefthalf of word # -1
      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
                                                                 0958
CASE csstkb OF                                                   0959
   >= csstkx:                                                    0960
      BEGIN                                                      0961
      IF anode.addr THEN                                         0962
         CASE prules OF                                          0963
            <= 0:      % no entries in table %                   0964
               IF ckaltcnt( anode.addr ) > nmalt THEN            0965
                  BEGIN  % too many paths %                      0966
                  IF ckoptnode( anode.addr ) THEN                0967
                     BEGIN  % entire rule optional %            01116
                     IF astr.L <= locstr.M THEN                 01117
                        *locstr* _ *astr*                       01118
                     ELSE                                       01119
                        BEGIN
                        &savstr _ getstring(astr.L, $dspblk);
                                                                01120
                        *savstr* _ *astr*;                      01121
                        END;                                    01122
                     nprint( &astr, IF anode.nsuccessor THEN
                     anode.nsuccessor ELSE altsuc, TRUE,
                     FALSE);                                     0969
                     IF NOT &savstr THEN                        01130
                        *astr* _ *locstr*                       01131
                     ELSE                                       01132
                        BEGIN                                   01133
                        *astr* _ *savstr*;                      01134
                        freestring(&savstr := 0, $dspblk);
```

```
                                                                    01135
                                                                    01136
            END;                                                     0971
         END;                                                        0972
      % get rule name to output string %                             0973
         gtruln( anode.addr, $locstr);                               0974
         *astr* _ *astr*, *locstr*, SP;                              0975
      % make entry in post-processing stack %                        0976
         appstk( anode.addr, 0 );                                    0977
      RETURN( FALSE );                                               0978
      END                                                            0979
   ELSE
      BEGIN  % expand rule: recursive call to
      nprint %                                                       0980
      IF astr.L <= locstr.M THEN                                     01137
         *locstr* _ *astr*                                           01138
      ELSE                                                           01139
         BEGIN                                                       01140
         &savstr _ getstring(astr.L, $dspblk);
                                                                     01141
         *savstr* _ *astr*;                                          01142
         END;                                                        01143
      nprint( &astr, anode.addr, TRUE,                               0982
         IF anode.nsuccessor THEN anode.nsuccessor
                                                                     0983
         ELSE altsuc);                                               0984
      IF NOT &savstr THEN                                            01144
         *astr* _ *locstr*                                           01145
      ELSE                                                           01146
         BEGIN                                                       01147
         *astr* _ *savstr*;                                          01148
         freestring(&savstr := 0, $dspblk);                          01149
         END;                                                        01150
      % 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
                                                                     0997
                  RETURN(
                     [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( anode.addr ) THEN
```

```
                                                                    01002
                                    % if entire rule optional,
                                    make pass without rule %        01003
                                       BEGIN                        01004
                                       IF astr.L <= locstr.M THEN
                                                                    01151
                                       *locstr* _ *astr*            01152
                                       ELSE                         01153
                                       BEGIN                        01154
                                       &savstr _ getstring(astr.L,
                                       $dspblk);                    01155
                                       *savstr* _ *astr*;           01156
                                       END;                         01157
                                       nprint(                      01006
                                       &astr, IF anode.nsuccessor
                                       THEN anode.nsuccessor ELSE
                                       altsuc, TRUE, FALSE);        01007
                                       IF NOT &savstr THEN          01158
                                       *astr* _ *locstr*            01159
                                       ELSE                         01160
                                       BEGIN                        01161
                                       *astr* _ *savstr*;           01162
                                       freestring(&savstr := 0,
                                       $dspblk);                    01163
                                       END;                         01164
                                       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
                                                                    01019
                        END;                                        01020
                  RETURN( TRUE );                                   01021
                  END;                                              01022
               ENDCASE
                  BEGIN  % expand rule: recursive call to nprint %  01023
                  IF astr.L <= locstr.M THEN                        01165
                     *locstr* _ *astr*                              01166
                                                                    01167
                  ELSE                                              01168
                     BEGIN                                          01169
                     &savstr _ getstring(astr.L, $dspblk);          01170
                     *savstr* _ *astr*;                             01171
                     END;                                           01025
                  nprint( &astr, anode.addr, TRUE,                  01026
                     IF anode.nsuccessor THEN anode.nsuccessor      01027
                     ELSE altsuc);                                  01172
                  IF NOT &savstr THEN                               01173
                     *astr* _ *locstr*                              01174
                  ELSE                                              01175
                     BEGIN
```

```
                *astr* _ *savstr*;                                      01176
                freestring(&savstr := 0, $dspblk);                      01177
                END;                                                    01178
             % 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* _ CR, LF, *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* _ CR, LF, CR, LF, *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
      ( anode );                                                               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 anode;                                                                  0455
                                                                               0456
   CASE anode.opcode OF                                                        0457
      = $option:                                                               0458
         IF &anode # anode.alternative THEN                                    0459
            IF &anode _ anode.alternative THEN REPEAT CASE                     0460
            ELSE RETURN( TRUE )                                                0461
         ELSE RETURN( TRUE );                                                  0462
      = $xecute:                                                               0463
         IF ckoptnode( anode.addr ) THEN                                       0464
            IF &anode # anode.alternative THEN                                 0465
               IF &anode _ anode.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
                                                                                0474
   % FORMALS %                                                                  0475
      ( anode );                                                                0476
   % ALGORITHM %
      % 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
                                                                                0479
   LOCAL i;                                                                     0480
   REF anode;                                                                   0481
                                                                                0482
   i _ 0;                                                                       0483
   WHILE &anode DO                                                              0484
      BEGIN                                                                     0485
      IF anode.nsuccessor THEN                                                  0486
         i _ i - 1 + ckaltcnt( anode.nsuccessor );                              0487
            % subtract one so we dont count this node twice %                   0488
      BUMP i;                                                                   0489
      IF &anode = anode.alternative THEN EXIT LOOP;                             0490
      &anode _ anode.alternative;                                               0491
      END;                                                                      0492
   RETURN( i );
   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 tp1;                                              0501
   REF astr;                                                            0502
                                                                        0503
   ddgtsymbol( &astr, raddr);                                           0504
   IF FIND SF(*astr*) '[ [']] ^tp1 THEN                                 0505
      *astr* _ tp1 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, pfcall, 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, savstr;                                               0547
    LOCAL STRING locstr[200];                                               0548
    REF astr, anode, savstr;                                                0549
                                                                            0550
    &savstr _ 0;                                                           01179
    CASE [anode.nsuccessor].opcode OF                                       0551
      = $confirm: RETURN( FALSE );                                          0552
      =$keyop, =$vwspecs, =$levadj, =$option, =$pfcall, =$xecute,
      =$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
             IF astr.L <= locstr.M THEN                                 01180
                *locstr* _ *astr*                                       01181
             ELSE                                                       01182
                BEGIN                                                   01183
                &savstr _ getstring(astr.L, $dspblk);                   01184
                *savstr* _ *astr*;                                      01185
                END;                                                    01186
             *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
             IF NOT &savstr THEN                                        01187
                *astr* _ *locstr*                                       01188
             ELSE                                                       01189
                BEGIN                                                   01190
                *astr* _ *savstr*;                                      01191
                freestring(&savstr := 0, $dspblk);                      01192
                END;                                                    01193
             *astr* _ *astr*, "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
```

```
(prbuglook) PROCEDURE  % lookbug parsefunction handler %                0611
                                                                        0612
   % FORMALS %                                                          0613
      (astr, % adr of string to be appended to %                        0614
      anode, % adr of tree to be interpretted %                         0615
      altsuc);       % if TRUE, adr of tree to continue with           0616
         after exhausting anode tree %                                  0617
   % ALGORITHM %
      % 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
                                                                        0625
   LOCAL tsuc, talt, savstr;                                            0626
   LOCAL STRING locstr[200];                                            0627
   REF astr, anode, savstr;                                             0628
                                                                       01194
   &savstr _ 0;                                                         0629
   CASE [anode.nsuccessor].opcode OF                                    0630
      = $confirm: RETURN( FALSE );
      =$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
                IF astr.L <= locstr.M THEN                               01195
                   *locstr* _ *astr*                                     01196
                ELSE                                                     01197
                   BEGIN                                                 01198
                   &savstr _ getstring(astr.L, $dspblk);                 01199
                   *savstr* _ *astr*;                                    01200
                   END;                                                  01201
                *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
                IF NOT &savstr THEN                                      01202
                   *astr* _ *locstr*                                     01203
                ELSE                                                     01204
                   BEGIN                                                 01205
                   *astr* _ *savstr*;                                    01206
                   freestring(&savstr := 0, $dspblk);                    01207
                   END;                                                  01208
                *astr* _ *astr*, "BUG ";                                 0670
                nprint(                                                  0671
                   &astr, [anode.nsuccessor].alternative, TRUE,          0672
                   altsuc);                                              0673
                RETURN( TRUE );                                          0674
                END;                                                     0675
          ENDCASE                                                        0676
             CASE [anode.nsuccessor].alternative OF                      0677
                = 0, = anode.nsuccessor:                                 0678
                   BEGIN                                                 0679
                   &anode _ anode.nsuccessor;                            0680
                   REPEAT CASE 2;                                        0681
                   END;                                                  0682
                ENDCASE                                                  0683
                   BEGIN                                                 0684
                   *astr* _ *astr*, "BUG ";                              0685
                   RETURN( FALSE );                                      0686
                   END;
END.                                                                     0687
                                                                         0688
%%
```

```
(prnumlook) PROCEDURE  % looknum parsefunction handler %                    0689
                                                                            0690
   % FORMALS %                                                              0691
      (astr, % adr of string to be appended to %                            0692
      anode, % adr of tree to be interpretted %                             0693
      altsuc);       % if TRUE, adr of tree to continue with               0694
         after exhausting anode tree %                                      0695
   % ALGORITHM %
      % 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
                                                                            0702
   LOCAL tsuc, talt, savstr;                                                0703
   LOCAL STRING locstr[200];                                                0704
   REF astr, anode, savstr;                                                 0705
                                                                           01209
   &savstr _ 0;                                                             0706
   CASE [anode.nsuccessor].opcode OF
      =$confirm, =$keyop, =$vwspecs, =$levadj, =$option, =$pfcall,          0707
      =$xecute, =$call:                                                     0708
         BEGIN                                                              0709
         *astr* _ *astr*, "NUMBER ";                                        0710
         RETURN( FALSE );                                                   0711
         END;                                                               0712
      = $ssel, = $lsel, = $dsel:                                            0713
         CASE [anode.nsuccessor].alternative OF                             0714
            = 0, = &anode:                                                  0715
               BEGIN                                                        0716
               *astr* _ *astr*, "NUMBER ";                                  0717
               &anode _ anode.nsuccessor;                                   0718
               nprint(                                                      0719
                  &astr,                                                    0720
                  IF anode.nsuccessor                                       0721
                     THEN anode.nsuccessor                                  0722
                     ELSE altsuc,                                           0723
                  TRUE,                                                     0724
                  IF anode.nsuccessor THEN altsuc ELSE FALSE                0725
                  );                                                        0726
               RETURN( TRUE );                                              0727
               END;                                                         0728
            ENDCASE                                                         0729
               BEGIN
```

```
            IF astr.L <= locstr.M THEN                                  01210
               *locstr* _ *astr*                                        01211
            ELSE                                                        01212
               BEGIN                                                    01213
               &savstr _ getstring(astr.L, $dspblk);                    01214
               *savstr* _ *astr*;                                       01215
               END;                                                     01216
            *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 onode( &astr );                                         0745
            IF NOT &savstr THEN                                         01217
               *astr* _ *locstr*                                        01218
            ELSE                                                        01219
               BEGIN                                                    01220
               *astr* _ *savstr*;                                       01221
               freestring(&savstr := 0, $dspblk);                       01222
               END;                                                     01223
            *astr* _ *astr*, "NUMBER ";                                  0746
            nprint(                                                      0747
               &astr, [anode.nsuccessor].alternative, TRUE,              0748
               altsuc);                                                  0749
            RETURN( TRUE );                                              0750
            END;                                                         0751
      ENDCASE                                                            0752
         CASE [anode.nsuccessor].alternative OF                          0753
            = 0, = anode.nsuccessor:                                     0754
               BEGIN                                                     0755
               &anode _ anode.nsuccessor;                                0756
               REPEAT CASE 2;                                            0757
               END;                                                      0758
            ENDCASE                                                      0759
               BEGIN                                                     0760
               *astr* _ *astr*, "NUMBER ";                               0761
               RETURN( FALSE );                                          0762
               END;                                                      0763
END.                                                                    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, savstr;                                               0773
   LOCAL STRING locstr[200];                                             0774
   REF astr, anode, savstr;                                              0775
                                                                         0776
   &savstr _ 0;                                                         01224
   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
            IF astr.L <= locstr.M THEN                                  01225
               *locstr* _ *astr*                                        01226
            ELSE                                                        01227
               BEGIN                                                    01228
               &savstr _ getstring(astr.L, $dspblk);                    01229
               *savstr* _ *astr*;                                       01230
               END;                                                     01231
            nprint( &astr, IF anode.nsuccessor THEN anode.nsuccessor
            ELSE altsuc, TRUE, FALSE);                                   0787
            IF NOT &savstr THEN                                         01232
               *astr* _ *locstr*                                        01233
            ELSE                                                        01234
               BEGIN                                                    01235
               *astr* _ *savstr*;                                       01236
               freestring(&savstr := 0, $dspblk);                       01237
               END;                                                     01238
            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
         IF astr.L <= locstr.M THEN                                      01239
            *locstr* _ *astr*                                            01240
         ELSE                                                            01241
            BEGIN                                                        01242
            &savstr _ getstring(astr.L, $dspblk);                        01243
```

```
            *savstr* _ *astr*;                                          01244
            END;                                                        01245
         nprint( &astr, anode.addr, TRUE,                               0800
            IF anode.nsuccessor THEN anode.nsuccessor                   0801
            ELSE altsuc);                                               0802
         IF NOT &savstr THEN                                            01246
            *astr* _ *locstr*                                           01247
         ELSE                                                           01248
            BEGIN                                                       01249
            *astr* _ *savstr*;                                          01250
            freestring(&savstr := 0, $dspblk);                          01251
            END;                                                        01252
         % 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 savstr;                                                   01253
   LOCAL STRING locstr[200];                                       01082
   REF astr, anode, savstr;                                        01083
                                                                   01254
   &savstr _ 0;                                                    01255
   IF astr.L <= locstr.M THEN                                      01256
      *locstr* _ *astr*                                            01257
   ELSE                                                            01258
      BEGIN                                                        01259
      &savstr _ getstring(astr.L, $dspblk);                        01260
      *savstr* _ *astr*;                                           01261
      END;                                                         01262
   nprint( &astr, anode.addr, TRUE,                                01085
      IF anode.nsuccessor THEN anode.nsuccessor                    01086
      ELSE altsuc);                                                01087
   IF NOT &savstr THEN                                             01263
      *astr* _ *locstr*                                            01264
   ELSE                                                            01265
      BEGIN                                                        01266
      *astr* _ *savstr*;                                           01267
      freestring(&savstr := 0, $dspblk);                           01268
      END;                                                         01269
   RETURN( TRUE );                                                 01089
   END.
                                                                   01090
                                                                   01091
                                                                   01270
                                                                   01271
                                                                   01272
                                                                   01273
                                                                   01274
FINISH                                                              0812
```

C SYNTAX

```
< NLS, CSYNTAX.NLS;5, >, 16-SEP-74 12:12 EKM ;;;;
F'LE calculator % CML <NLS>calculator %% (CML,)
(PROGRAMS,calculator.cml,) %                                                02
% COMMON RULES %                                                           03
   % ENTITY DEFINITIONS %                                                  04
      editentity = textent / structure;                                     05
   % TEXT ENTITY DEFINITIONS %                                             06
      textent = text1 / "TEXT"!L1! / "LINK"!L1!;                            07
      text1 = "CHARACTER"!L1! / "WORD"!L1! / "VISIBLE"!L1! /
      "INVISIBLE"!L1! / "NUMBER"!L1!;                                       08
   % STRUCTURE ENTITY DEFINITIONS %                                        09
      structure = "STATEMENT"!L1! / notstatement;                           010
      notstatement = "GROUP"!L1! / "BRANCH"!L1! / "PLEX"!L1! ;              011
   % SWITCH %                                                              012
      switch = ("ON"!L1 1!/"OFF"!2!);                                       013
   % DEVICE TYPES %                                                        014
      devopt =                                                              015
         ( "TASKER"                                                         016
         / "TI"!L1! <"Terminal">                                            017
         / "NVT"!L1!                                                        018
         / "LINEPROCESSOR"!L1!                                              019
         / "IMLAC"!L1!                                                      020
         / "EXECUPORT"!L1!                                                  021
         / "33-TTY"!L1 33!                                                  022
         / "35-TTY"!L1 35!                                                  023
         / "37-TTY"!L1 37! );                                               024
% DECLARATIONS %                                                           025
    DECLARE PARSEFUNCTION                                                   026
      answ,              % reads answer construct %                        027
      answer,            % for questions - returns  0/1 %                  028
      sp,                % reads next char, TRUE if space %                029
      readconfirm,       % reads next char if ca %                         030
      readbug,  % reads next char if BUG %                                 031
      readoption,        % TRUE if next char is optchar %                  032
      readrepeat,        % TRUE if next char is repeat %                   033
      lookansw, % TRUE if next char is Y/CA %                              034
      lookconfirm,       % TRUE if next char is CA/REPEAT/INSERT %         035
      lookbug,  % TRUE if next char is BUG %                               036
      looknum,  % TRUE if next char is a number %                          037
      clearname,         % clears the name area %                          038
      notca;             % reads next char, TRUE iff not CA char %         039
    DECLARE EXTERNAL zinsstatement;                                         040
    DECLARE EXT-KEYWORD % so only one copy exists in system These
    keywords are defined as external strings in CONST. %                    041
       % STRUCTURAL ENTITIES %                                             042
          "BRANCH",                                                         043
          "GROUP",                                                          044
          "PLEX",                                                           045
          "STATEMENT",                                                      046
       % TEXTUAL ENTITIES %                                               047
          "CHARACTER",                                                      048
          "INVISIBLE",                                                      049
          "LINK",                                                           050
          "NUMBER",                                                         051
          "PASSWORD",                                                       052
          "TEXT",                                                           053
```

```
        "VISIBLE",                                                      054
        "WORD",                                                         055
      % MISC. ENTITIES %                                                056
        "FILE",                                                         057
        "OLDFILELINK",                                                  058
        "NEWFILELINK",                                                  059
        "NAME",                                                         060
        "RETURN",                                                       061
        "FILERETURN",                                                   062
        "EDGE",                                                         063
        "MARKER";                                                       064
   DECLARE EXTERNAL % not defined here see (nls,pdata,) %               065
      alisubs,                                                          066
      nlssubs;                                                          067
% CALCULATOR SUBSYSTEM COMMANDS %                                      068
   SUBSYSTEM calculator KEYWORD "CALCULATOR"                            069
      INITIALIZATION                                                    070
         zcinit=                                                        071
            xcalcinit();                                                072
      TERMINATION                                                       073
         zcterm=                                                        074
            xcquit();                                                   075
      RENTRY                                                            076
         zclcrent =                                                     077
            xcreenter();                                                078
                                                                        079
      COMMAND zcshow =                                                  080
         "SHOW"                                                         081
         ("ACCUMULATOR"!L1!  <"Registers">                              082
            CONFIRM                                                     083
            xcshoaccums()                                               084
         /"FILE" !L1 NOTT!<"in window">                                 085
            DSEL(#"EDGE")                                               086
            CONFIRM                                                     087
            xcshofil()                                                  088
         );                                                             089
      COMMAND zadd =                                                    090
         ("ADD"!L1! / "+"!L1! )                                         091
            param2 _ LSEL(#"NUMBER")                                    092
            CONFIRM                                                     093
            %  execute ADD (param2)    %                              094
            xcarith(#"ADD",param2)                                      095
            cfeedback();                                                096
      COMMAND zcclear =                                                 097
         "CLEAR"!L1!                                                    098
            ("ACCUMULATOR" !L1!                                         099
               CONFIRM                                                 0100
               xclraccum()                                             0101
            /"FILE" !L1!                                               0102
               CONFIRM                                                 0103
               xclrfil()                                               0104
            );                                                         0105
      COMMAND zdivide =                                                0106
         ("DIVIDE"!L1! / "/"!L1! )                                     0107
            param2 _ LSEL(#"NUMBER")                                   0108
            CONFIRM                                                    0109
```

```
        %  execute DIVIDE (param2)   %                                  0110
        xcarith(#"DIVIDE",param2)                                       0111
        cfeedback();                                                    0112
                                                                        0113
   COMMAND zevaluate =                                                  0114
      "EVALUATE"                                                        0115
         param _ LSEL(#"TEXT")                                          0116
         xceval(param)                                                  0117
         (("ADD"!L1!  / "+"!L1! / CONFIRM)                              0118
            param _ #"ADD"                                              0119
         /("DIVIDE"!L1!  / "/"!L1!)                                     0120
            param _ #"DIVIDE"                                           0121
         /("MULTIPLY"!L1!  / "*"!L1!  / "X"!L1!)                        0122
            param _ #"MULTIPLY"                                         0123
         /("SUBTRACT"!L1!   / "-"!L1!)                                  0124
            param _ #"SUBTRACT" )                                       0125
         CONFIRM                                                        0126
         xcevend(param)                                                 0127
         cfeedback();                                                   0128
   COMMAND zcformat =                                                   0129
      "FORMAT"!L1!                                                      0130
         ("PLACES"!L1! <"to the">                                       0131
            param _                                                     0132
            ("RIGHT"!L1!                                                0133
            /"LEFT"!L1!                                                 0134
            )                                                           0135
            param2 _ LSEL(#"NUMBER")                                    0136
               CONFIRM                                                  0137
               xfdigits(param,param2)                                   0138
         /"COMMAS"!L1!                                                  0139
            param _ answer()                                            0140
            CONFIRM                                                     0141
            xcomma(param)                                               0142
         /param _                                                       0143
            ("LEFT"!L1!                                                 0144
            /"RIGHT"!L1!)                                               0145
            <"justify">                                                 0146
            CONFIRM                                                     0147
            xcjust(param)                                               0148
         /"TERSE"!L1 NOTD! <"output">                                   0149
            param _ answer()                                            0150
            CONFIRM                                                     0151
            xcfeedb(param)                                              0152
         /"DOLLAR"!L1! <"signs">                                        0153
            param _ answer()                                            0154
            CONFIRM                                                     0155
            xdollar(param)                                              0156
         ) cfeedback();                                                 0157
   COMMAND zcinsert =                                                   0158
      "INSERT"!L1!                                                      0159
         <"accum following">                                            0160
         level _ NULL                                                   0161
         (ent _ ( text1 / "LINK"!L1! )                                  0162
            dest _ DSEL(ent)                                            0163
         / ent _ "TEXT"!L1!                                             0164
            dest _ DSEL(#"CHARACTER")                                   0165
```

```
      / ent _ structure                                                0166
         dest _ DSEL(#"STATEMENT")                                     0167
         level _ LEVADJ)                                               0168
      CONFIRM                                                          0169
      %  execute INSERT ACCUM (dest, param)    %                     0170
      xinsert(ent, dest, level,xcinsac());                             0171
COMMAND zmultiply =                                                    0172
   ("MULTIPLY"!L1! / "*"!L1! / "X"!L1! )                               0173
      param2 _ LSEL(#"NUMBER")                                         0174
      CONFIRM                                                          0175
      %  execute cmult (param2)   %                                   0176
      xcarith(#"MULTIPLY",param2)                                      0177
      cfeedback();                                                     0178
COMMAND zsubtract =                                                    0179
   ("SUBTRACT"!L1! / "-"!L1! )                                         0180
      param2 _ LSEL(#"NUMBER")                                         0181
      CONFIRM                                                          0182
      %  execute SUBTRACT (param2)   %                                0183
      xcarith(#"SUBTRACT",param2)                                      0184
      cfeedback();                                                     0185
COMMAND zcreplace =                                                    0186
   "REPLACE"!L1!                                                       0187
      dent _ editentity <"at">                                         0188
      dest _ DSEL (dent)                                               0189
      sent _ #"TEXT"                                                   0190
      <"by accumulator">                                               0191
      CONFIRM                                                          0192
      %  execute replace    %                                         0193
      xreplace(dent,dest,sent,xcinsac());                              0194
COMMAND ztotal =                                                       0195
   "TOTAL"!L1!                                                         0196
      CONFIRM                                                          0197
      xctot1()                                                         0198
      ;                                                                0199
COMMAND zuse =                                                         0200
   "USE"!L1!                                                           0201
      ("ACCUMULATOR" !L1! <"number">                                   0202
         param2 _ LSEL(#"NUMBER")                                      0203
         CONFIRM                                                       0204
         xcuseaccum(param2)                                            0205
         cfeedback()                                                   0206
      /"SAVED" !L1! <"Accumulators">                                   0207
         CONFIRM                                                       0208
         xcusesaved()                                                  0209
      );                                                               0210
COMMAND zwrite =                                                       0211
   "WRITE"!L1!  <"new">                                                0212
      "FILE"!L1!                                                       0213
      param _ LSEL(#"NEWFILELINK")                                     0214
      CONFIRM                                                          0215
      xcwritef(param)                                                  0216
   ;                                                                   0217
COMMAND znumber=                                                       0218
   looknum() %see if its another number %                              0219
   param2 _ LSEL(#"NUMBER")                                            0220
   CONFIRM                                                             0221
```

```
        xcarith(#"ADD",param2)                                  0222
        cfeedback();                                            0223
                                                                0224
                                                                0225
    END.                                                        0226
FINISH OF CALCLANGUAGE
```