ADRMNP

```
< NLS, ADRMNP.NLS.14, >, 25-Apr-78 13:13 JDH ;;;;< NLS, ADRMNP.NLS;41,
>, 18-JUN-74 13:21 KEV ;
FILE adrmnp  % 110  <rel-nls>adrmnp % % (110,)  (rel-nls,adrmnp.rel,)  %
                                                                        0175
  % DUCUMENTATION %                                                     01655
    % the following is the formal description of a link that is used
    by the procedures that parse links:                                 01656
      s  :=  [SP / TAB / CR / LF / EOL]  /  s (SP / TAB / CR / LF /
      EOL)                                                              01657
      link  :=  opndlm  s  body  s  clsdlm                              01658
        opndlm  :=  ( ('< / '() [comment] )  /  "--"                    01659
        clsdlm  :=  ')  /  '>                                           01660
        comment  :=  ctxt "--"                                          01661
        ctxt  :=  any number of characters excluding the following:
                                                                        01662
          '- / ', / '> / ') / '< / '( / '. / '; / ': /                  01663
          '! / '* / '# / '/ / '\ / '" / '' / '= / '+                    01664
      body  :=  [filspc]  [dae]  ['': vwspc]                            01665
      filspc  :=  [[hn ',] un ',] fn ',                                 01666
      hn  :=  s hstnam s  /  NULL                                       01667
        hstnam  :=  1$48(LD/'-)                                         01668
      un  :=  s usrnam s                                                01669
        usrnam  :=  zero to 39 characters excluding the following
                                                                        01670
          ', / SP / TAB / EOL / ': / '; /                               01671
          '< / '> / '= / '_ / '* / '@ / '. /                            01672
          [0B,5B] / [7B,32B] / [34B,36B] / 140B / >= 173B               01673
      fn  :=  s (filnam / filnam2) s                                    01674
        filnam  :=  zero or more characters excluding the following
                                                                        01675
          ', / SP / TAB / EOL / ': /                                    01676
          '< / '> / '= / '_ / '@ /                                      01677
          [0B,5B] / [7B,32B] / [34B,36B] / 140B / >= 173B               01678
        filnam2  :=  '= delim1 STRING delim2                            01679
        delim1  :=  CHARACTER                                           01680
        delim2  :=  the same character used for delim                   01681
      dae  :=  element  /  dae element                                  01682
      element  :=                                                       01683
        s  /                                                            01684
        stmntnam  /  '! stmntnam  /  '* stmntnam  /                     01685
        '# marker  /                                                    01686
        '/  /  '\  /                                                    01687
        ('" STRING '"  /  '' CHAR) s ['= search]  /                     01689
        '. pelement  /  '+ selement  /  '- selement                     01690
      marker  :=                                                        01691
        a letter followed by any number of characters except the
        following:                                                      01692
          SP / TAB / ') / '> / ':                                       01693
      stmntnam  :=                                                      01694
        LD  [LD / '- / '' / '@]  /  stmntnam  (LD / '- / '' / '@)
                                                                        01695
      search  :=  stype [sdomain]  /  sdomain [stype]                   01696
      stype  :=  [NUMBER] ('C  /  'W)                                   01697
        (can be upper or lower case letters)                            01698
      sdomain  :=  [NUMBER] 'S                                          01699
        (can be upper or lower case letters)                            01700
```

```
selement  :=  [NUMBER] struc  /  selement [NUMBER] struc                01702
struc  :=  'C  /  'E  /  'F  /  'I  /  'L  /  'N  /  'V  /  'W
                                                                        01703
   (can be upper or lower case letters)                                 01704
pelement  :=  [NUMBER] pos  /  pelement [NUMBER] pos                    01705
pos  :=                                                                 01706
   'B  /  'C  /  'D  /  'E  /  "FR"  /  'H  /  'L  /                    01707
   'N  /  'O  /  'P  /  'R  /  'S  /  'T  /  'U  /  'W                  01708
      (can be upper or lower case letters)                              01709
vwspc  :=  [viewspec]  /  vwspc viewspec                                01710
viewspec  :=                                                            01711
    s  /                                                                01712
   'a  /  'b  /  'c  /  'd  /  'e  /  'f  /  'g  /  'h  /               01713
   'i  /  'j  /  'k  /  'l  /  'm  /  'n  /  'o  /  'p  /               01714
   'q  /  'r  /  's  /  't  /  'u  /  'v  /  'w  /  'x  /               01715
   'y  /  'z  /  'A  /  'B  /  'C  /  'D  /  'G  /  'H  /               01716
   'I  /  'J  /  'K  /  'L  /  'O  /  'P  /  filter                     01717
filter  :=  '; CONTENT-PATTERN ';                                       01718
%                                                                       01720
```

```
REGISTER r1 = 1, r2 = 2, r3 = 3;                                    01231
% Link field extraction. %                                         05974
   (lnkpspc)  %extract fields from link and setup link data block%
                                                                    05502
      PROCEDURE                                                     05503
         (lnknum,              %link number%                        05504
         tptadr,               %t-ptr from which to start link search%
                                                                    05505
         usrstg,               %string for directory name%          05506
         fnmstg,               %string for file name%               05507
         addexpstg,  %string for address expression%                05508
         vspstg,               %string for viewspecs%               05509
         adstr);               % addres of data block to be filled %
                                                                    05510
      LOCAL                                                         05511
         ind,                                                       05512
         rhstn;       % host number %                               05513
      LOCAL TEXT POINTER                                            05514
         tp1, tp2, tp3, tp4, tp5, tp6, tp7, tp8;                    05515
      REF tptadr, usrstg, fnmstg, addexpstg, vspstg, adstr;         05516
                                                                    05517
      % find and parse the right link and get the file name %       05518
         IF &tptadr THEN                                            05519
            BEGIN                                                   05520
            tp8 _ tptadr;  tp8[1] _ tptadr[1];                      05521
            ind _ 1;                                                05522
            DO                                                      05523
               BEGIN                                                05524
               rhstn _ lnbfls( $tp8, &adstr, &fnmstg);              05525
               CASE ind OF                                          05526
                  = 1: NULL;                                        05527
                  ENDCASE                                           05528
                     IF adstr[ls+1] <= tp8[1] THEN                  05529
                        BEGIN                                       05530
                        SIGNAL( lnk5err, $"Illegal Link:
                        Left Delimiter Not Found");                 05531
                        END;                                        05532
               tp7 _ adstr[ls];  tp7[1] _ adstr[ls+1];              05533
               tp8 _ adstr[le];  tp8[1] _ adstr[le+1];              05534
               BUMP ind;                                            05535
               END UNTIL ind > lnknum;                              05536
            END                                                     05537
         ELSE                                                       05538
            BEGIN                                                   05539
            rhstn _ lnbfls( 0, &adstr, &fnmstg );                   05540
            tp7 _ adstr[ls];  tp7[1] _ adstr[ls+1];                 05541
            END;                                                    05542
      % set up other strings %                                      05543
         tp1 _ adstr[us];  tp1[1] _ adstr[us+1];                    05544
         tp2 _ adstr[ue];  tp2[1] _ adstr[ue+1];                    05545
         *usrstg* _ + tp1 tp2; % directory %                        05546
         tp3 _ adstr[ds];  tp3[1] _ adstr[ds+1];                    05547
         tp4 _ adstr[de];  tp4[1] _ adstr[de+1];                    05548
         *addexpstg* _ tp3 tp4; % address expression %              05549
         tp5 _ adstr[vb];  tp5[1] _ adstr[vb+1];                    05550
         tp6 _ adstr[ve];  tp6[1] _ adstr[ve+1];                    05551
```

```
   *vspstg* _ tp5 tp6; % view %                                    05552
RETURN( rhstn );                                                   05553
END.
                                                                   05554
%  %                                                              05555
```

```
(lnbfls) % create file name string from link parse block%   03409
   PROCEDURE                                                03410
      (stp,       % FALSE or t-ptr for use for parsing a link %
                                                           03411
      adstr,      % address of parsed link data structure or zero
      %                                                    03412
      astr);      % address of string to get file name %   03413
   LOCAL                                                   03414
      i,  % index for copying data structures %            03415
      rhstn,      % cell to get remote host number %        03416
      ldstr[40];  % local data structure for link parsing % 03417
   LOCAL TEXT POINTER                                      03418
      h1, h2, u1, u2, f1, f2;                              03419
   LOCAL STRING                                            03420
      locstr[200];                                         03421
   REF stp, adstr, astr;                                   03422
                                                           03423
   % RETURNS %                                            03424
      % will generate error if invalid hostname field in link.
      Otherwise, will return filled in string that can be used for
      gtjfn and will return hostnumber (number for local host if
      none specified in the link %                         03425
                                                           03426
   % ALGORITHM %                                          03427
      % for local hosts the following is how the file string is
      built up:                                            03428
         if a user name is specified (that doesn't end with a
         control-f or an altmode) then the file name is the
         concatenation of:                                 03429
            '< specified-user-name '> specified-file-name  03430
               if the file name is not specified, it is set to the
               file currently loaded (or generates an error for
               typed in links)                             03431
         if a user name is specified (that ends with a control-f
         or an altmode) then the file name is the concatenation
         of:                                               03432
            '< specified-user-name specified-file-name     03433
               if the file name is not specified, it is set to the
               file currently loaded (or generates an error for
               typed in links)                             03434
         if no user name is specified for a typed in link, then
         the file name string is merely the specified file name
         (or NULL if no file name specified)               03435
         if no user name is specified for a bugged link, then the
         file name string consists of the concatenation of: 03436
            '< default-directory '> specified-file-name    03437
               if the file name is not specified, it is set to the
               file currently loaded                       03438
      for remote hosts, if a user name is specified, then the same
      algorithm is used as is used for the local host.     03439
      for remote hosts without a user name, the returned string is
      the specified file name (or generates an error if no file
      name specified for a typed in link)                  03440
      %                                                    03441
                                                           03442
   % parse link or copy data structure as necessary %      03443
```

```
        IF &stp THEN                                                    03444
           BEGIN                                                        03445
           lnkprs( &stp, $ldstr);                                       03446
           IF &adstr THEN                                               03447
              FOR i _ 1 UP UNTIL > lnkdsl DO adstr[i-1] _
              ldstr[i-1];                                               03448
           END                                                          03449
        ELSE                                                            03450
           IF &adstr THEN                                               03451
              FOR i _ 1 UP UNTIL > lnkdsl DO ldstr[i-1] _ adstr[i-1]
                                                                        03452
           ELSE err( $"NLS System Error: Illegal Call to LNBFLS");
                                                                        03453
     % initialize text pointers %                                       03454
        h1 _ ldstr[hs];  h1[1] _ ldstr[hs+1];                           03455
        h2 _ ldstr[he];  h2[1] _ ldstr[he+1];                           03456
        u1 _ ldstr[us];  u1[1] _ ldstr[us+1];                           03457
        u2 _ ldstr[ue];  u2[1] _ ldstr[ue+1];                           03458
        f1 _ ldstr[fs];  f1[1] _ ldstr[fs+1];                           03459
        f2 _ ldstr[fe];  f2[1] _ ldstr[fe+1];                           03460
     % pick up any specified user name %                                03461
        *locstr* _ + u1 u2;                                             03462
     % act according to whether its a local host or not %              03463
        CASE (rhstn _ gthstn( $h1, $h2 ) ) OF                           03464
           < 0:  err($"invalid host name");                             03465
           = lhostn:        % local hosts %                             03466
              (lcl):                                                    03467
              CASE locstr.L OF                                          03468
                 = 0:       % no user name specified %                  03469
                    BEGIN                                               03470
                    *locstr* _ f1 f2;                                   03471
                    CASE ldstr[lfn] OF                                  03472
                       = 0: % typed in link %                           03473
                          IF locstr.L THEN *astr* _ *locstr*            03474
                          ELSE *astr* _ NULL;                           03475
                       ENDCASE      % bugged link %                     03476
                          BEGIN                                         03477
                          IF NOT gdftdir( ldstr[lfn], &astr) THEN
                                                                        03478
                             BEGIN %pass it if it's a journal item%
                                                                        06605
                             IF FIND f1 D THEN *astr* _ NULL            06607
                             ELSE err ($"illegal username in file
                             header");                                  06604
                             END                                        06606
                          ELSE *astr* _ '<, *astr*, '>;                 03479
                          IF locstr.L THEN *astr* _ *astr*, *locstr*
                                                                        03480
                          ELSE lnbafn( &astr, ldstr[lfn] );             03481
                          END;                                          03482
                    END;                                                03483
                 ENDCASE    % user name specified %                     03484
                    BEGIN                                               03485
                    CASE *locstr*[locstr.L] OF                          03486
                       = '<^F>, = '<ESC>:                               03487
                          *astr* _ '<, *locstr*;                        03488
```

```
                    ENDCASE                                          03489
                       *astr* _ "<, *locstr*, ">;                    03490
                 *locstr* _ f1 f2;                                   03491
                 CASE locstr.L OF                                    03492
                    = 0:                                             03493
                       IF ldstr[lfn] THEN lnbafn( &astr,
                       ldstr[lfn] )                                  03494
                       ELSE err($"Illegal Link");                    03495
                    ENDCASE                                          03496
                       *astr* _ *astr*, *locstr*;                    03497
                 END;                                                03498
        ENDCASE  % remote hosts %                                    03499
           IF locstr.L THEN GOTO lcl                                 03500
           ELSE                                                      03501
              BEGIN                                                  03502
              *astr* _ f1 f2;                                        03503
              IF NOT astr.L THEN                                     03504
                 IF NOT ldstr[lfn] THEN err($"Illegal Link")
                                                                     03505
                 ELSE                                                03506
                    lnbafn( &astr, ldstr[lfn] );                     03507
              END;                                                   03508
% now return %                                                       03509
   RETURN( rhstn );                                                  03510
END.
                                                                     03511
%  %                                                                03512
```

```
(lnbafn)  % append file name string to passed string %          03513
   PROCEDURE                                                    03514
      (astr,       % string to be appended to %                 03515
      fileno);     % file number for file name string %         03516
   LOCAL                                                        03517
      ldstr[40];  % data structure to parse link %              03518
   LOCAL TEXT POINTER                                           03519
      tp1, tp2, tp3;                                            03520
   LOCAL STRING locstr[200];                                    03521
   REF astr;                                                    03522
                                                                03523
   % get the file name %                                        03524
      filnam( fileno, $locstr);                                 03525
   % now parse the link %                                       03526
      FIND SF(*locstr*) ^tp3;                                   03527
      lnkprs( $tp3, $ldstr);                                    03528
      tp1 _ ldstr[fs];  tp1[1] _ ldstr[fs+1];                   03529
      tp2 _ ldstr[fe];  tp2[1] _ ldstr[fe+1];                   03530
   % do the appending and return %                              03531
      *astr* _ *astr*, tp1 tp2;                                 03532
      RETURN;                                                   03533
   END.
                                                                03534
   %  %                                                        03535
```

```
(lnkprs)  % detailed link parser %                                    04929
   PROCEDURE                                                         04930
      (stp,       % address of text pointer to start scan %          04931
      adstr       % address of  linkparams  data strcture %          04932
      );                                                             04933
   LOCAL rflag;                                                      04934
   LOCAL TEXT POINTER tp1, tp2;                                      04935
   REF stp, adstr;                                                   04936
                                                                     04937
   % the following signals (maintained in CONST) can be generated
   while parsing a link:                                             04938
      lnk1err -                                                      04939
         Illegal Link Syntax:
         ENDCHR Before Closing "                                     04940
      lnk2err -                                                      04941
         Illegal Link Syntax:
         ENDCHR Before CH After '                                    04942
      lnk3err -                                                      04943
         Illegal Link Syntax:
         Illegal Statement Name or Number After ! or *               04944
      lnk4err -                                                      04945
         Illegal Link Syntax:
         Illegal DAE Element                                         04946
      lnk5err -                                                      04947
         Illegal Link:
         Left Delimeter Not Found                                    04948
      lnk6err -                                                      04949
         Illegal Link Syntax or Semantic:
         Missing Right Delimeter or Bad Viewspecs                    04950
      lnk7err -                                                      04951
         Illegal Link Syntax:
         Missing Right Delimeter                                     04952
      lnk8err -                                                      04953
         Illegal Link Syntax:
         Illegal Marker After #                                      04954
      lnk9err -                                                      04955
         Illegal Link Syntax:
         ENDCHR Before ; In Filter                                   04956
      %                                                              04957
                                                                     04958
   % initialize %                                                    04959
      rflag _ TRUE;                                                  04960
   % setup working t-ptr %                                          04961
      FIND stp ^tp1 ^tp2;                                            04962
      IF FIND > '- THEN FIND < $'- ^tp1 ^tp2;                        04963
   % setup default directory file number %                          04964
      adstr[1fn] _ stp.stfile;                                       04965
   % try different starting points if errors %                      04966
      ON SIGNAL                                                      04967
         # lnk5err:                                                  04968
            IF rflag AND (adstr[ls+1] >= tp1[1]) THEN                04969
               BEGIN                                                 04970
               tp1[1] _ adstr[cs+1];                                 04971
               GOTO lnkps1;                                          04972
               END                                                   04973
            ELSE                                                     04974
```

```
                BEGIN                                                   04975
                rflag _ FALSE;                                          04976
                tp1[1] _ MIN(tp1[1], tp2[1], adstr[ls+1]);              04977
                IF FIND tp1 < [".( / "< / "--"] ^tp1 >                  04978
                   THEN GOTO lnkps1;                                    04979
                END;                                                    04980
        ELSE;                                                           04981
(lnkps1):                                                               04982
% find start of the link %                                              04983
   lnkstr( $tp1, &adstr);                                               04984
% find an optional comment %                                           04985
   lnkcom( &adstr );                                                    04986
% find the body of the link %                                          04987
   lnkbdy( &adstr );                                                    04988
% find the right delimeter %                                           04989
   lnkend( &adstr );                                                    04990
% done, so return %                                                    04991
   RETURN;                                                              04992
                                                                        04993
END.                                                                    
                                                                        04994
%  %                                                                   04995
```

```
(lnkstr) %***%    % find the start of a link %                          06031
   PROCEDURE                                                            06032
      (stp,       % address of text pointer to start scanninfg
      from %                                                            06033
      adstr       % address of  linkparams  data strcture %             06034
      );                                                                06035
   LOCAL lnktyp;                                                        06036
   LOCAL TEXT POINTER tp1;                                              06037
   REF stp, adstr;                                                      06038
                                                                        06039
   % ABNORMAL RETURNS %                                                 06040
      % can generate the following signal:                              06041
         lnk5err:                                                       06042
            Illegal Link:
            Left Delimeter Not Found                                    06043
      %                                                                 06044
                                                                        06045
   % position to scan starting text pointer %                           06046
      FIND stp > ^tp1;                                                  06047
      IF NOT stp.stastr THEN                                            06048
         BEGIN                                                          06049
         tp1[1] _ MAX( tp1[1], fchtxt(stp) );                           06050
         FIND tp1 >;                                                    06051
         END;                                                           06052
   % initially normal type link %                                       06053
      lnktyp _ FALSE;                                                   06054
   LOOP                                                                 06055
      CASE READC OF                                                     06056
         = "(, = "<:                                                    06057
            BEGIN                                                       06058
            FIND ^tp1 _tp1;                                             06059
            EXIT LOOP;                                                  06060
            END;                                                        06061
         = "-:                                                          06062
            IF FIND 1$"- < 2CH ^tp1 THEN                                06063
               BEGIN                                                    06064
               lnktyp _ TRUE;                                           06065
               EXIT LOOP;                                               06066
               END;                                                     06067
         = ENDCHR, = "), = ">:                                          06068
            BEGIN                                                       06069
            FIND stp <;                                                 06070
            LOOP                                                        06071
               CASE READC OF                                            06072
                  = "(, = "<:                                           06073
                     BEGIN                                              06074
                     FIND ^tp1;                                         06075
                     EXIT LOOP 2;                                       06076
                     END;                                               06077
                  = "-:                                                 06078
                     IF FIND "- ^tp1 THEN                               06079
                        BEGIN                                           06080
                        lnktyp _ TRUE;                                  06081
                        EXIT LOOP 2;                                    06082
                        END;                                            06083
                  = ENDCHR:                                             06084
```

```
                    BEGIN                                                   06085
                    tp1[1] _ 0;                                             06086
                    EXIT LOOP 2;                                            06087
                    END;                                                    06088
                  ENDCASE;                                                  06089
            END;                                                            06090
         ENDCASE;                                                           06091
   IF (NOT tp1[1]) OR (NOT stp.stastr) THEN                                 06092
      IF (NOT tp1[1]) OR (tp1[1]<fchtxt(stp)) THEN                          06093
         BEGIN                                                              06094
         SIGNAL( lnk5err, $"Illegal Link:
         Left Delimiter Not Found");                                        06095
         END;                                                               06096
   adstr[ls] _ tp1;  adstr[ls+1] _ tp1[1];                                  06097
   % initialize to no comment %                                             06098
      adstr[cs] _ adstr[ce] _ adstr[ls];                                    06099
      adstr[cs+1] _ adstr[ce+1] _ adstr[ls+1]+1;                            06100
   % no comment if link starts with "--" %                                  06101
      IF lnktyp THEN                                                        06102
         BEGIN                                                              06103
         % make comment live after "--" %                                   06104
            adstr[cs+1] _ adstr[ce+1] _ adstr[ls+1] + 2;                    06105
         END;                                                               06106
   RETURN;                                                                  06107
   END.
                                                                            06108
   %  %                                                                     06109
```

```
(lnkcom)  % find the optional comment in a link %                    0226
   PROCEDURE                                                         0227
      (adstr      % address of  linkparams  data strcture %          0228
      );                                                             0229
   LOCAL TEXT POINTER tp1;                                           0238
   REF adstr;                                                        0230
                                                                     0231
   % initialize %                                                    0257
      tp1 _ adstr[cs];                                               0234
      tp1[1] _ adstr[cs+1];                                          0235
   % no comment if link starts with "--" %                           0239
      IF (adstr[ce+1] - adstr[cs+1]) = 2 THEN RETURN;                0232
   % setup to scan forward from start of link looking for comment
   %                                                                 0240
      FIND tp1 >;                                                    0243
   % scan until "--" or any other link delimeter %                   0244
      LOOP                                                           0245
         CASE READC OF                                               0246
            = '-:                                                    0247
               IF FIND 1$'- ^tp1 THEN                                0261
                  BEGIN                                              0262
                  adstr[ce] _ tp1;                                   0263
                  adstr[ce+1] _ tp1[1];                              0264
                  RETURN;                                            0265
                  END                                                0266
               ELSE RETURN;                                          0267
            = ENDCHR,                                                0255
            = ',, = '>, = '), = '<, = '(, = '.,                      0248
            = ';, = ':, = '!, = '*, = '#, = '/,                      0253
            = '\, = '", = '', = '=, = '+:                            0254
               RETURN;                                               0260
            ENDCASE;                                                 0256
                                                                     0270
   END.
                                                                     0271
   %  %                                                             0249
```

```
(lnkbdy)  % find the body of a link %                                  01573
   PROCEDURE                                                          01574
      (adstr      % address of  linkparams  data strcture %           01575
      );                                                             01576
   LOCAL TEXT POINTER stp;                                           01577
   REF adstr;                                                        01578
                                                                     01579
   % find the optional filspc part of the link %                     01580
      lnkfpc( &adstr, $stp );                                        01581
   % find the optional filspc part of the link %                     01582
      lnkdae( &adstr, $stp );                                        01583
   % find the optional vwspc part of the link %                      01584
      lnkvws( &adstr, $stp );                                        01585
   % done, so return %                                              01586
      RETURN;                                                        01587
                                                                     01588
   END.
                                                                     01589
   %  %                                                             01590
```

```
(lnkIpc)  % find the optional filspc of a link %                    0377
   PROCEDURE                                                        0378
      (adstr,     % address of  linkparams  data strcture %        0379
      tp1 % address of text pointer to get end of filspc %          0999
      );                                                            0382
   REF adstr, tp1;                                                  0380
                                                                    0381
   % find an optional hostname %                                    0272
      lnkhst( &adstr, &tp1 );                                       0273
   % find an optional username %                                    0332
      lnkusr( &adstr, &tp1 );                                       0333
   % find an optional filename %                                    0459
      lnkfil( &adstr, &tp1 );                                       0460
   % done so return %                                              0391
      RETURN;                                                       0392
                                                                    0393
   END.
                                                                    0394
   %  %                                                            0395
```

```
(lnkhst)  % find the optional hostname in a link %                    0334
   PROCEDURE                                                         0335
      (adstr,     % address of  linkparams  data strcture %          0336
      stp % text pointer gets end of hostname field %                01002
      );                                                             0337
   LOCAL TEXT POINTER tp1, tp2;                                      0338
   REF adstr, stp;                                                   0339
                                                                     0345
   % initialize to no hostname %                                     0346
      adstr[hs] _ adstr[he] _ tp1 _ adstr[ce];                       0347
      adstr[hs+1] _ adstr[he+1] _ tp1[1] _ adstr[ce+1];              0348
   % setup scan forward from end of comment (skip spaces or tabs)
   %                                                                 0349
      FIND tp1 > $(SP/TAB/CR/LF/EOL) ^tp1 ^stp;                     0350
   % hostname is 0-48 letters, digits, or minus signs, followed by
   any number of spaces or tabs, and terminated by a comma %         0351
      IF FIND $48(LD/'-) ^tp2 $(SP/TAB/CR/LF/EOL) ', ^stp THEN
                                                                     0352
         BEGIN                                                       0353
         adstr[hs] _ tp1;                                            0354
         adstr[hs+1] _ tp1[1];                                       0355
         adstr[he] _ tp2;                                            0356
         adstr[he+1] _ tp2[1];                                       0357
         RETURN;                                                     0359
         END                                                         0360
      ELSE RETURN;                                                   0361
                                                                     0362
   END.
                                                                     0363
   %  %                                                             0364
```

```
(inkusr)  % find the optional username in a link %                    0461
   PROCEDURE                                                         0462
      (adstr,     % address of  linkparams  data strcture %          0463
      stp % text ptr to start scan; gets end of username field %
                                                                     0464
      );                                                             0466
   LOCAL                                                             0962
      i    % character count index %                                 0963
      ;                                                              0964
   LOCAL TEXT POINTER tp1, tp2;                                      0467
   REF adstr, stp;                                                   0468
                                                                     0475
   % initialize to no username %                                     0476
      adstr[us] _ adstr[ue] _ adstr[he];                             0477
      adstr[us+1] _ adstr[ue+1] _ adstr[he+1];                       0478
   % setup scan forward from end of hostname (skip spaces & tabs)
   %                                                                 0479
      FIND stp > $(SP/TAB/CR/LF/EOL) ^tp1 ^stp;                     0480
   % find username if it exists, else make hostname username and
   make hostname null %                                              0484
      FOR i _ 1 UP UNTIL > 39 DO                                     0485
         CASE READC OF                                               0487
            % illegal username characters %                         0488
            = ",, = SP, = TAB, = CR, = LF, = EOL:                    0489
               BEGIN                                                 0490
               IF NOT FIND < CH ^tp2 > $(SP/TAB/CR/LF/EOL) ",
               ^stp THEN                                             0491
                  GOTO notusr;                                       0492
               adstr[us] _ tp1;  adstr[us+1] _ tp1[1];               0493
               adstr[ue] _ tp2;  adstr[ue+1] _ tp2[1];               0494
               RETURN;                                              01003
               END;                                                  0506
            = ":, = ";, = "<, = ">, = "=, = "_, = "*, = "@,
                                                                     0507
            = ENDCHR,                                                0508
            IN [0B,5B] % ^@ - ^E %,                                  0509
            IN [7B,32B]          % ^G - ^Z %,                         0510
            IN [34B,36B]         % ^] - ^^ %,                         0511
            = 140B, >= 173B:                                         0512
               (notusr):       BEGIN                                 0513
               % make username field = hostname field %              0514
                  adstr[us] _ adstr[hs];                             0515
                  adstr[us+1] _ adstr[hs+1];                         0516
                  adstr[ue] _ adstr[he];                             0517
                  adstr[ue+1] _ adstr[he+1];                         0518
               % make hostname field null %                          0519
                  adstr[he] _ adstr[hs];                             0520
                  adstr[he+1] _ adstr[hs+1];                         0521
               RETURN;                                               0522
               END;                                                  0523
         ENDCASE;                                                    0524
      % more than 39 characters is not a user name %                 0960
         GOTO notusr;                                                0961
                                                                     0525
   END.
                                                                     0526
```

% %

```
(lnkfil)  % find the optional filename in a link %                    0274
   PROCEDURE                                                         0275
      (adstr,     % address of  linkparams  data strcture %          0276
      stp % text ptr to start scan; gets ent of filename field %
                                                                     0366
      );                                                             0277
   LOCAL                                                            01026
      char        % delimeter character %                           01027
      ;                                                             01028
   LOCAL TEXT POINTER tp1, tp2;                                     0278
   REF adstr, stp;                                                  0279
                                                                    0320
   % initialize to no filename %                                    0281
      adstr[fs] _ adstr[fe] _ adstr[ue];                            0282
      adstr[fs+1] _ adstr[fe+1] _ adstr[ue+1];                      0283
   % setup scan forward from end of username (skip spaces & tabs)
   %                                                                0291
      FIND stp > $(SP/TAB/CR/LF/EOL) ^tp1 ^stp;                    0294
   % find filename if it exists, else make username filename and
   make username hostname and make hostname null %                  0396
      CASE READC OF                                                 01004
         = ':=:                                                     01005
            CASE char _ READC OF                                    01009
               = ENDCHR:  GOTO notfil;                              01010
               ENDCASE                                              01011
                  BEGIN                                             01017
                  FIND ^tp1;                                        01018
                  LOOP                                              01012
                     CASE READC OF                                  01013
                        = ENDCHR:  GOTO notfil;                     01014
                        = char:                                     01015
                           IF FIND < CH ^tp2 > CH
                           $(SP/TAB/CR/LF/EOL) ', ^stp              01020
                              THEN GOTO isfil                       01022
                           ELSE GOTO notfil;                        01021
                        ENDCASE;                                    01016
                  END;                                              01019
         = ENDCHR: GOTO notfil;                                     03654
         ENDCASE                                                    01006
            BEGIN                                                   01333
            FIND < CH >;                                            01335
            LOOP                                                    0398
               CASE READC OF                                        0402
                  % illegal filename characters %                   0418
                     = ',, = SP, = TAB, = CR, = LF, = EOL:          0436
                        BEGIN                                       0438
                        IF NOT FIND < CH ^tp2 >
                        $(SP/TAB/CR/LF/EOL) ', ^stp                 0437
                           THEN GOTO notfil;                        01007
                        (isfil):                                    01023
                           adstr[fs] _ tp1;  adstr[fs+1] _ tp1[1];
                                                                    0440
                           adstr[fe] _ tp2;  adstr[fe+1] _ tp2[1];
                                                                    0441
                           RETURN;                                  01008
                        END;                                        0445
```

```
                    = ':, = '<, = '>, = '=, = '_, = '@,                    0413
                    = ENDCHR,                                              0419
                    IN [0B,5B]   % ^@ - ^E %,                              0414
                    IN [7B,25B]  % ^G - ^U %,                              0415
                    IN [27B,32B] % ^W - ^Z %,                             06799
                    IN [34B,36B] % ^] - ^^ %,                              0416
                    = 140B, >= 173B:                                       0417
                       (notfil): BEGIN                                     0422
                       % make filename field = username field %
                                                                           0451
                          adstr[fs] _ adstr[us];                           0423
                          adstr[fs+1] _ adstr[us+1];                       0452
                          adstr[fe] _ adstr[ue];                           0424
                          adstr[fe+1] _ adstr[ue+1];                       0453
                       % make username field = hostname field %
                                                                           0528
                          adstr[us] _ adstr[hs];                           0529
                          adstr[us+1] _ adstr[hs+1];                       0530
                          adstr[ue] _ adstr[he];                           0531
                          adstr[ue+1] _ adstr[he+1];                       0532
                       % make hostname field null %                       0454
                          adstr[he] _ adstr[hs];                           0425
                          adstr[he+1] _ adstr[hs+1];                       0455
                       RETURN;                                             0428
                       END;                                                0429
                 ENDCASE;                                                  0420
           END;                                                           01334
                                                                           0314
     END.
                                                                           0315
     % %                                                                  0316
```

```
(lnkdae)  % find the optional dae in a link %                          01339
   PROCEDURE                                                           01340
      (adstr,     % address of  linkparams  data strcture %           01341
      stp % text pointer to start scan; gets end of dae %              01342
      );                                                               01343
   LOCAL                                                               01344
      lnkpar[29]  % additional link params block for recursion %
                                                                       01345
      ;                                                                01346
   LOCAL TEXT POINTER tp1, tp2, tp3;                                   01347
   REF adstr, stp;                                                     01351
                                                                       01352
   % ABNORMAL RETURNS %                                               01353
      % this procedure can generate the following signals:            01354
         lnk2err:                                                      01357
            Illegal Link Syntax:
            ENDCHR Before CH After '                                   01358
      %                                                                01363
                                                                       01364
   % initialize to no dae %                                           01365
      (daeprs):                                                        04486
      adstr[ds] _ adstr[de] _ adstr[fe];                               01366
      adstr[ds+1] _ adstr[de+1] _ adstr[fe+1];                         01367
   % setup scan forward from end of filename (skip spaces & tabs)
   %                                                                   01368
      FIND stp > $(SP/TAB/CR/LF/EOL) ^tp1 ^stp;                        01369
   % find dae if it exists %                                          01370
      LOOP                                                             01371
         CASE READC OF                                                 01372
            = ':                                                       01732
               BEGIN                                                   01733
               FIND ^stp < CH $(SP/TAB/CR/LF/EOL) ^tp2;                01734
               (lkgdae):                                               01735
                  IF tp2[1] < tp1[1] THEN tp2[1] _ tp1[1];             01736
                  adstr[ds] _ tp1;  adstr[ds+1] _ tp1[1];              01737
                  adstr[de] _ tp2;  adstr[de+1] _ tp2[1];              01738
                  RETURN;                                              01739
               END;                                                    01740
            = '), = '>:                                                01741
               BEGIN                                                   01742
               FIND < CH $(SP/TAB/CR/LF/EOL) ^tp2 ^stp;                01743
               GOTO lkgdae;                                            01744
               END;                                                    01745
            = ":                                                       01753
               LOOP                                                    01754
                  CASE READC OF                                        01755
                     = ", = ENDCHR:  EXIT LOOP;                        01762
                     ENDCASE;                                          01781
            = ':                                                       01782
               CASE READC OF                                           01783
                  = ENDCHR:                                            01784
                     BEGIN                                             01785
                     SIGNAL( lnk2err, $"Illegal Link Syntax:
                     ENDCHR Before CH After '");                       01788
                     END;                                              01789
                  ENDCASE;                                             01790
```

```
        = "(, = "<, ="",, = ENDCHR: % some common bad chrs %
                                                                    04467
           IF adstr[fe+1] > adstr[fs+1] THEN  % backup %            04920
              BEGIN                                                 04921
              % make filename field = username field %              04470
                 adstr[fs] _ adstr[us];                             04471
                 adstr[fs+1] _ adstr[us+1];                         04472
                 stp _ adstr[fe] _ adstr[ue];                       04473
                 stp[1] _ adstr[fe+1] _ adstr[ue+1];                04474
              % make username field = hostname field %              04475
                 adstr[us] _ adstr[hs];                             04476
                 adstr[us+1] _ adstr[hs+1];                         04477
                 adstr[ue] _ adstr[he];                             04478
                 adstr[ue+1] _ adstr[he+1];                         04479
              % make hostname field null %                         04480
                 adstr[he] _ adstr[hs];                             04481
                 adstr[he+1] _ adstr[hs+1];                         04482
              % now reparse the dae %                              04483
                 FIND stp > $(SP/TAB/CR/LF/EOL) ", ^stp;            04487
                 GOTO daeprs;                                       04485
              END                                                   04923
           ELSE RETURN;                                             04922
        ENDCASE;                                                    01894
                                                                    01537
END.
                                                                    01538
% %                                                                 01539
```

```
(lnkvws)  % find the optional vwspc in a link %                    0871
   PROCEDURE                                                       0872
      (adstr,     % address of  linkparams  data strcture %       0873
      stp % text pointer to start scan; gets end of dae %         01001
      );                                                          0876
   LOCAL TEXT POINTER tp1, tp2;                                   0877
   REF adstr, stp;                                                0878
                                                                  01272
   % ABNORMAL RETURNS %                                           01549
      % this routine can generate the following signal:           01550
         lnk6err:                                                 01551
            Illegal Link Syntax or Semantic:
            Missing Right Delimeter or Bad Viewspecs              01552
         lnk9err:                                                 01642
            Illegal Link Syntax:
            ENDCHR Before ; In Filter                             01643
      %                                                           01555
                                                                  01554
   % initialize to no vwspc %                                     0886
      adstr[vb] _ adstr[ve] _ adstr[de];                          0887
      adstr[vb+1] _ adstr[ve+1] _ adstr[de+1];                    0888
   % setup scan forward from end of dae (skip spaces & tabs) %
                                                                  0889
      FIND stp > $(SP/TAB) ^tp1 ^stp;                             0890
   % find vwspc if it exists %                                    0894
      LOOP                                                        01030
         CASE READC OF                                            01031
            =SP, =TAB, =CR, =LF, =EOL, IN['a,'z], IN['A,'Z]:      01540
               NULL;                                              01564
            = '), ='>, = ENDCHR:                                  01541
               BEGIN                                              01565
               FIND < CH $(SP/TAB/CR/LF/EOL) ^tp2;                01566
               IF tp2[1] < tp1[1] THEN tp2[1] _ tp1[1];           01646
               adstr[vb] _ tp1;  adstr[vb+1] _ tp1[1];            01567
               adstr[ve] _ tp2;  adstr[ve+1] _ tp2[1];            01568
               RETURN;                                            01569
               END;                                               01570
            = ';:                                                 01542
               LOOP                                               01558
                  CASE READC OF                                   01559
                     = ';:  EXIT LOOP;                            01560
                     = '':                                        01628
                        READC;                                    01629
                     = '":                                        01630
                        LOOP                                      01632
                           CASE READC OF                          01633
                              = '": EXIT LOOP;                    01634
                              = ENDCHR: EXIT LOOP;                01635
                              ENDCASE;                            01636
                     = ENDCHR:                                    01561
                        BEGIN                                     01637
                        SIGNAL( lnk9err, $"Illegal Link Syntax:
                        ENDCHR Before ; In Filter");              01640
                        END;                                      01641
                     ENDCASE;                                     01562
               ENDCASE                                            01543
```

```
                BEGIN                                                   01544
                SIGNAL( lnk6err, $"Illegal Link Syntax or
                Semantics:
                Missing Right Delimiter or Bad Viewspecs");             01547
                END;                                                    01548
                                                                         0934
        END.
                                                                         0935
        %  %                                                            0936
```

```
(lnkend)  % find the right delimeter of a link %                    0838
   PROCEDURE                                                        0839
      (adstr      % address of  linkparams  data strcture %         0840
      );                                                            0841
   LOCAL TEXT POINTER tp1;                                         01000
   REF adstr;                                                       0847
                                                                    0848
   % ABNORMAL RETURNS %                                            01601
      % this routine can generate the following signals:            01602
         lnk7err:                                                   01603
            Illegal Link Syntax:
            Missing Right Delimeter                                 01604
      %                                                             01605
                                                                    01606
   % setup to scan from end of viewspecs %                         01591
      tp1 _ adstr[ve];  tp1[1] _ adstr[ve+1];                       01592
   % find the end of the link %                                     01593
      IF NOT FIND tp1 > $(SP/TAB/CR/LF/EOL) (')/">) ^tp1 THEN
                                                                    01594
         BEGIN                                                      01595
         SIGNAL( lnk7err, $"Illegal Link Syntax:
         Missing Right Delimiter");                                 01598
         END;                                                       01599
   % now set up the text pointer in the linkparams block %         01609
      adstr[le] _ tp1;  adstr[le+1] _ tp1[1];                       01610
   % done, so return %                                             0842
      RETURN;                                                       0843
                                                                    0844
   END.
                                                                    0845
   %  %                                                           0846
```

```
%Address Expression evaluation%                                        01948
   (caddexp)   % GB: core NLS Address Expression evaluation % %***%
   PROCEDURE (t1, t2, da, ptr % => vs1, vs2, ca, useq, rring %);
                                                                       06563
      % Procedure description                                         06564
         FUNCTION                                                     06565
            Given two text pointers to a text string, a pointer to a
            display area record (for viewspecs and return rings), and
            a text pointer into a file, this routine will evaluate
            the expression relative to the text pointer and update
            the text pointer.  It will also return updated viewspec
            words, content analyzer program, and user sequence
            generator program.  It will also return 0 or the address
            of the relevant statement return ring if the last thing
            done is a .fr .                                          06566
            Note: Viewspecs in effect (e.g. d, i, 0) are IGNORED in
            computing the new address.                               06592
         ARGUMENTS                                                   06567
            t1 - TEXT POINTER - start of address expression string
                                                                     06568
            t2 - TEXT POINTER - end of address expression string
                                                                     06582
            da - DISPLAY AREA - for viewspecs and return rings       06583
            ptr - TEXT POINTER -                                     06584
               base point from which to compute the new address
               (updated during the computation)                      06585
         RESULTS                                                     06569
            procedure-value - INTEGER - first viewspec word          06570
            2nd-value - INTEGER - second viewspec word               06586
            3rd-value - PROCEDURE ADDRESS -                          06587
               content analyzer procedure                            06590
            4th-value - PROCEDURE ADDRESS -                          06588
               user sequence generator procedure                     06591
            5th-value - RING - address of file return ring           06589
         NON-STANDARD CONTROL                                        06571
            A variety of errors can be generated.                    06572
         GLOBALS                                                     06573
            none                                                     06574
         EXAMPLE                                                     06575
            none                                                     06576
         %                                                           06577
      % Declarations %                                               06578
         LOCAL                                                       06114
            savslh,  % save value of slashflg %                      06115
            rhstn,   % host number from links %                      06116
            fnlsls,  % for doing final / %                           06117
            sdomain, % domain for content searches %                 06118
            savs1,   % saved viewspecs for errors %                  06119
            savs2,                                                   06593
            cacode,  % saved address of content analyzer program %
                                                                     06120
            useqgen, % saved address of sequence generator program %
                                                                     06121
            proc,                                                    06122
            dir,     % direction for scan relations %                 06123
            char,    % for main loop dae parsing %                    06124
```

```
        count,   % for scan and positional relationships %  06125
        scnbck,                                             06126
        sring,   % adr stmnt return ring for file return, etc. %
                                                            06127
        rsrng,   % adr stmnt ret ring to be returned %      06128
        fname,   % adr file return fing for file return %   06129
        tmp,                                                06130
        fl,      % file number for jump name external file %
                                                            06131
        enftype, % TRUE, look for ext name link in file origin;
        FALSE, go to user-options only%                     06608
        ldstr[35];       % data structure for link parsing %
                                                            06132
     LOCAL STRING                                           06133
        jnestr[200],     % for jump name external file %    06134
        locstr[200],     % to hold dae locally %            06135
        st1[100], st2[100], st3[100], st4[300]; %for link
        parsing%                                            06136
     LOCAL TEXT POINTER                                     06137
        pscan, tp1, tp2, tp9, z1;                           06138
     REF t1, t2, da, ptr, fl;                               06139
  % initialize %                                            06141
     savslh _ slashflg := FALSE;                            06142
     fnlsls _ rsrng _ FALSE;                                06143
  % save value of viewspec data %                           06144
     savs1 _ da.davspec;                                    06145
     savs2 _ da.davspc2;                                    06146
     cacode _ da.dacacode;                                  06147
     useqgen _ da.dausqcod;                                 06148
  % cleanup on errors %                                     06149
     ON SIGNAL ELSE                                         06150
        BEGIN                                               06151
        slashflg _ savslh;                                  06152
        da.davspec _ savs1;                                 06153
        da.davspc2 _ savs2;                                 06154
        da.dacacode _ cacode;                               06155
        da.dausqcod _ useqgen;                              06156
        END;                                                06157
  % copy dae to minimize page faulting, etc. %             06158
     *locstr* _ t1 t2;                                      06159
  FIND SF(*locstr*) ^z1 ^tp9 >;                             06160
  LOOP                                                      06161
     BEGIN                                                  06162
     IF inptrf THEN GOTO caeerror;                          06163
     count _ 1;                                             06164
     FIND z1 >;                                             06165
     *errwrk* _ char _ READC;                               06166
     FIND ^z1;                                              06167
     CASE char OF                                           06168
        = SP, = TAB, = EOL, = CR, = LF: REPEAT LOOP;        06169
        = ENDCHR: NULL;                                     06170
        ENDCASE fnlsls _ rsrng _ FALSE;                     06171
     CASE char OF                                           06172
        = SP, = TAB, = EOL, = CR, = LF:  NULL;              06173
        ='.:   % positional relation %                      06174
           LOOP                                             06175
```

```
BEGIN                                                          06176
FIND z1 >;                                                     06177
*errwrk* _ char _ READC;                                       06178
FIND ^z1;                                                      06179
count _ 1;                                                     06180
CASE char OF                                                   06181
   IN ['0, '9]:                                                06182
      BEGIN                                                    06183
      FIND < CH >;                                             06184
      count _ gadnum();                                        06185
      *errwrk* _ char _ READC;                                 06186
      FIND ^z1;                                                06187
      CASE char OF                                             06188
         #L: GOTO caeerror;                                    06189
         ENDCASE REPEAT CASE 2(char);                          06190
      END;                                                     06191
   ='B, ='b:  % back %  %SHOULD USE SEQGEN%                    06192
      BEGIN                                                    06193
      rsrng _ FALSE;                                           06194
      getpr($getbck, count, &ptr);                             06195
      END;                                                     06196
   ='C, ='c:  % next occurrence of content %                   06197
      BEGIN                                                    06198
      rsrng _ FALSE;                                           06199
      *errwrk* _ '", *conreg*, '";                             06200
      tmp _ 1;                                                 06201
      srchtype _ contnt;                                       06202
      FOR tmp UP UNTIL > count DO                              06203
         BEGIN                                                 06204
         IF (tmp > 1) AND (conreg.L = 1) THEN BUMP
         ptr[1];                                               06205
         specreg( $conreg, contnt, &ptr);                      06206
         END;                                                  06207
      END;                                                     06208
   ='D, ='d:  % down %                                         06209
      BEGIN                                                    06210
      rsrng _ FALSE;                                           06211
      getpr($getsub, count, &ptr);                             06212
      END;                                                     06213
   ='E, ='e:  % end %                                          06214
      BEGIN                                                    06215
      rsrng _ FALSE;                                           06216
      ptr _ getvnd(ptr, 1000 %large # of levels%);
                                                               06217
      ptr[1] _ 1;                                              06218
      END;                                                     06219
   ='F, ='f:  % file return %                                  06220
      BEGIN                                                    06221
      *errwrk* _ char _ READC;                                 06222
      CASE char OF                                             06223
         ='R, ='r:  NULL;                                      06224
         ENDCASE GOTO caeerror;                                06225
      FIND ^z1;                                                06226
      fname _ readfrring(da.dalink, count : rsrng);
                                                               06227
      FIND SF(*[fname]*) ^tp1;                                 06228
```

```
            IF (rhstn _ lnbfls($tp1, 0, $jnestr)) #
            lhostn THEN                                    06229
               err($"Remote File Manipulation Not
               Implemented Yet");                          06230
            ptr _ readsrring(rsrng, 0 : ptr[1],
            da.davspec, da.davspc2);                       06231
            ptr.stfile _ cloafil($jnestr);                 06232
            END;                                           06233
         ='H, ='h:  % head %                               06234
            BEGIN                                          06235
            rsrng _ FALSE;                                 06236
            getpr($gethed, count, &ptr);                   06237
            END;                                           06238
         ='L, ='l:  % indirect link %                      06239
            BEGIN                                          06240
            rsrng _ FALSE;                                 06241
            ptr[1] _ MAX(fchtxt(ptr), ptr[1]);             06242
            IF (rhstn _ lnkpspc(                           06243
               count, &ptr, $st1, $st2, $st3, $st4,
               $ldstr)) # lhostn THEN                      06244
                  err($"Remote File Manipulation Not
                  Implemented Yet");                       06245
            IF ldstr[fe+1] > ldstr[fs+1] THEN %filename%
                                                           06246
               ptr _ nfstid ($st2, $st3, $st4, &da :
               ptr[1], da.davspec, da.davspc2,
               da.dacacode, da.dausqcod)                   06247
            ELSE %old file%                                06248
               IF st3.L THEN % address expression %        06249
                  BEGIN                                    06250
                  FIND SF(*st3*) ^tp1 SE(*st3*) ^tp2;
                                                           06251
                  da.davspec _ caddexp( $tp1, $tp2, &da,
                  &ptr : da.davspc2, da.dacacode,
                  da.dausqcod );                           06252
                  END;                                     06253
            IF st4.L # empty THEN feedlt(&da, $st4);
                                                           06254
            END;                                           06255
         ='N, ='n:  % next %  %SHOULD USE SEQGEN%          06256
            BEGIN                                          06257
            rsrng _ FALSE;                                 06258
            getpr($getnxt, count, &ptr);                   06259
            END;                                           06260
         ='O, ='o:  % origin %                             06261
            BEGIN                                          06262
            rsrng _ FALSE;                                 06263
            ptr.stpsid _ origin;                           06264
            ptr[1] _ 1;                                    06265
            END;                                           06266
         ='P, ='p:  % predecessor %                        06267
            BEGIN                                          06268
            rsrng _ FALSE;                                 06269
            getpr($getprd, count, &ptr);                   06270
            END;                                           06271
         ='R, ='r:  % return %                             06272
```

```
                    BEGIN                                          06273
                    rsrng _ FALSE;                                 06274
                    readfrring(da.dalink, 0 : sring);              06275
                    ptr _ readsrring(sring, count : ptr[1],
                    da.davspec, da.davspc2);                       06276
                    END;                                           06277
                 ='S, ='s:  % succ %                               06278
                    BEGIN                                          06279
                    rsrng _ FALSE;                                 06280
                    getpr($getsuc, count, &ptr);                   06281
                    END;                                           06282
                 ='T, ='t:  % tail %                               06283
                    BEGIN                                          06284
                    rsrng _ FALSE;                                 06285
                    getpr($getail, count, &ptr);                   06286
                    END;                                           06287
                 ='U, ='u:  % up %                                 06288
                    BEGIN                                          06289
                    rsrng _ FALSE;                                 06290
                    getpr($getup, count, &ptr);                    06291
                    END;                                           06292
                 ='W, ='w:  % next occurrence of word %            06293
                    BEGIN                                          06294
                    rsrng _ FALSE;                                 06295
                    *errwrk* _ '", *conreg*, '", "=w";             06296
                    tmp _ 1;                                       06297
                    srchtype _ wordtyp;                            06298
                    FOR tmp UP UNTIL > count DO                    06299
                       BEGIN                                       06300
                       IF (tmp > 1) AND (conreg.L = 1) THEN BUMP
                       ptr[1];                                     06301
                       specreg( $conreg, wordtyp, &ptr);           06302
                       END;                                        06303
                    END;                                           06304
                 # LD:   % end of structural relation %            06305
                    BEGIN                                          06306
                    CASE char OF                                   06307
                       = ENDCHR:  NULL;                            06308
                       ENDCASE FIND z1 < CH ^z1;                   06309
                    REPEAT LOOP 2;                                 06310
                    END;                                           06311
                 ENDCASE GOTO caeerror;                            06312
              END;                                                 06313
        = '-:    % scan relation or link %                        06314
           CASE READC OF                                           06315
              = '-:      % link %                                   06316
                 BEGIN                                             06317
                 FIND < CH >;                                      06318
                 GOTO cadlnk;                                      06319
                 END;                                              06320
              = ENDCHR:  GOTO caeerror;                            06321
              ENDCASE    % scan relation %                         06322
                 BEGIN                                             06323
                 FIND < CH >;                                      06324
                 GOTO cadscr;                                      06325
                 END;                                              06326
```

```
        ="+:   % scan relation %                                        06327
           (cadscr):                                                    06328
           BEGIN                                                        06329
           dir _ IF char = "+ THEN 1 ELSE -1;                           06330
           LOOP                                                         06331
              BEGIN                                                     06332
              FIND z1 >;                                                06333
              *errwrk* _ char _ READC;                                  06334
              FIND ^z1;                                                 06335
              count _ dir;                                              06336
              CASE char OF                                              06337
                 IN ["0, "9]:                                           06338
                    BEGIN                                               06339
                    FIND < CH >;                                        06340
                    count _ gadnum() * dir;                             06341
                    *errwrk* _ char _ READC;                            06342
                    FIND ^z1;                                           06343
                    CASE char OF                                        06344
                       #L: GOTO caeerror;                               06345
                       ENDCASE REPEAT CASE 2(char);                     06346
                    END;                                                06347
                 ="C, ="c: %character scan%                             06348
                    BEGIN                                               06349
                    count _ gaddir(&ptr, count);                        06350
                    UNTIL (count _ count-1) < 0 DO FIND CH;             06351
                    FIND ^ptr;                                          06352
                    END;                                                06353
                 ="E, ="e: %statement end%                              06354
                    FIND SE(ptr) < CH ^ptr;                             06355
                 ="F, ="f: %statement front%                            06356
                    FIND SF(ptr) ^ptr;                                  06357
                 ="I, ="i: %invisible scan%                             06358
                    BEGIN                                               06359
                    count _ gaddir(&ptr, count : scnbck);               06360
                    UNTIL (count _ count-1) < 0 DO                      06361
                       FIND $NP $PT;                                    06362
                    IF scnbck THEN FIND $NP;                            06363
                    FIND ^ptr;                                          06364
                    END;                                                06365
                 ="L, ="l: %link scan%                                  06366
                    BEGIN                                               06367
                    count _ gaddir(&ptr, count : scnbck);               06368
                    UNTIL (count _ count-1) < 0 DO                      06369
                       FIND CH ["( / "<];                               06370
                    FIND ^ptr;                                          06371
                    END;                                                06372
                 ="N, ="n: %number scan%                                06373
                    BEGIN                                               06374
                    count _ gaddir(&ptr, count : scnbck);               06375
                    UNTIL (count _ count-1) < 0 DO                      06376
                       BEGIN                                            06377
                       FIND $D $(NOT D) ^pscan;                         06378
                       % call number delimiter routine to find
                       end of number %                                  06379
                          ndr( $pscan, $tp1, $tp2 );                    06380
                       % set pointer to end of current number %
```

```
                                                                    06381
                        IF scnbck                                   06382
                           THEN FIND tp1                            06383
                           ELSE FIND tp2 $(NOT D);                  06384
                     END;                                           06385
                  FIND ^ptr;                                        06386
                  END;                                              06387
               ="V, ="v: %visible scan%                             06388
                  BEGIN                                             06389
                  count _ gaddir(&ptr, count : scnbck);             06390
                  UNTIL (count _ count-1) < 0 DO                    06391
                     FIND $PT $NP;                                  06392
                  IF scnbck THEN FIND $PT;                          06393
                  FIND ^ptr;                                        06394
                  END;                                              06395
               ="W, ="w: %word scan%                                06396
                  BEGIN                                             06397
                  count _ gaddir(&ptr, count : scnbck);             06398
                  UNTIL (count _ count-1) < 0 DO                    06399
                     FIND $LD $NLD;                                 06400
                  IF scnbck THEN FIND $LD;                          06401
                  FIND ^ptr;                                        06402
                  END;                                              06403
               # LD:   % end of scan relation %                     06404
                  REPEAT CASE 2(char);                              06405
            ENDCASE GOTO caeerror;                                  06406
            END;                                                    06407
      END;                                                          06408
   ="/:   % slash %                                                 06409
      BEGIN                                                         06410
      ccurcon( &ptr, $errwrk );                                     06411
      typeas( $errwrk );                                            06412
      fnlsls _ TRUE;                                                06413
      END;                                                          06414
   ="\:   % backslash %                                             06415
      cprista(ptr, &da, &da);                                       06416
   = "(, = "<:      %link %                                        06417
      (cadlnk):                                                     06418
      BEGIN                                                         06419
      FIND < CH ^z1 ^tp1;                                           06420
      lnkprs( $z1, $ldstr);                                         06421
      ldstr[lfn] _ t1.stfile; %because copied bugged link%
                                                                    06422
      IF (rhstn _ lnkpspc( 1, 0, $st1, $st2, $st3, $st4,
      $ldstr)) # lhostn THEN                                        06423
         err($"Remote File Manipulations Not Implemented
         Yet");                                                     06424
      %because copied bugged link%                                  06425
         IF (st3.L = empty) AND                                     06426
            t1.stfile AND (ldstr[fs+1] >= ldstr[fe+1]) THEN
                                                                    06427
               *st3* _ "0, STRING(getsid(t1)), "+,
               STRING(t1[1]), "c;                                   06428
      z1 _ ldstr[le];  z1[1] _ ldstr[le+1];                         06429
      *errwrk* _ tp1 z1;                                            06430
      IF ldstr[fe+1] > ldstr[fs+1] THEN  % file name %              06431
```

```
            ptr _ nfstid( $st2, $st3, $st4, &da, : ptr[1],
            da.davspec, da.davspc2, da.cacode, da.dausqcod)
                                                          06432
         ELSE                                             06433
            IF st3.L THEN % address expression %          06434
               BEGIN                                      06435
               FIND SF(*st3*) ^tp1 SE(*st3*) ^tp2;        06436
               da.davspec _ caddexp( $tp1, $tp2, &da, &ptr :
               da.davspc2, da.dacacode, da.dausqcod );    06437
               END;                                       06438
         IF st4.L THEN feedlt( &da, $st4);                06439
         END;                                             06440
      ="": %content search%                               06441
         BEGIN                                            06442
         FIND < CH ^tp1 > CH;                             06443
         *st1* _ NULL;                                    06444
         gdlit2( $st1, '");                               06445
         FIND ^z1;                                        06446
         *errwrk* _ tp1 z1;                               06447
         (srchpar):                                       06448
            *conreg* _ *st1*;                             06449
            srchtype _ conspt( $z1 : count, sdomain);     06450
            FIND ^z1;                                     06451
            *errwrk* _ tp1 z1;                            06452
            CASE sdomain OF                               06453
               = -1:                                      06454
                  BEGIN                                   06455
                  tmp _ 1;                                06456
                  FOR tmp UP UNTIL > count DO             06457
                     BEGIN                                06458
                     IF (tmp > 1) AND (st1.L = 1) THEN BUMP
                     ptr[1];                              06459
                     specreg( $st1, srchtype, &ptr);      06460
                     END;                                 06461
                  END;                                    06462
               ENDCASE                                    06463
                  BEGIN                                   06464
                  tmp _ 1;                                06465
                  FOR tmp UP UNTIL > count DO             06466
                     LOOP                                 06467
                        BEGIN                             06468
                        FIND ptr ^tp1;                    06469
                        IF (tmp > 1) AND (st1.L = 1) THEN BUMP
                        ptr[1];                           06470
                        specreg( $st1, srchtype, &ptr);   06471
                        IF ptr # endfil THEN EXIT LOOP;   06472
                        IF (sdomain _ sdomain-1) THEN     06473
                           ptr _ getnxt(tp1)              06474
                        ELSE EXIT LOOP 2;                 06475
                        END;                              06476
                  END;                                    06477
         END;                                             06478
      ='' %apostrophe% : %character search%               06479
         BEGIN                                            06480
         FIND < CH ^tp1 > CH;                             06481
         *st1* _ char _ READC;                            06482
```

```
        *errwrk* _ '', char;                                        06483
        FIND ^z1;                                                   06484
        GOTO srchpar;                                               06485
        END;                                                        06486
      ='#: % marker %                                               06487
        BEGIN                                                       06488
        *st1* _ NULL;                                               06489
        gdlit2($st1, SP);                                           06490
        FIND ^z1;                                                   06491
        *errwrk* _ *errwrk*, *st1*;                                 06492
        ptr _ lkmkr($st1, ptr.stfile : ptr[1]);                     06493
        FIND z1 >;                                                  06494
        END;                                                        06495
      = '$,    % external name %                                    06496
      = '*,    % next name %                                        06497
      = '!,    % name in current branch %                          06498
      = LD, = '', = '-, = '@, = '&, = '(, = '):    % name %
                                                                    06499
        BEGIN                                                       06500
        tmp _ ptr.stfile;     % save file no. for jump name
        external %                                                  06501
        IF z1[1] > 2 THEN *locstr*[z1[1]-2] _ SP;                   06502
        gadname(&ptr, $st2, (CASE char OF                           06503
           = '$: extname;                                           06504
           = '*: seqname;                                           06505
           = '!: braname;                                           06506
           ENDCASE nametyp));                                       06507
        FIND ^z1;                                                   06508
        IF char = '$ AND ptr = endfil THEN                          06509
           BEGIN                                                    06609
           enftype _ TRUE;  %look first in file origin%             06610
           CASE getenf(tmp, enftype, $jnestr) OF                    06611
              # 0:  %TRUE return%                                   06612
                 BEGIN                                              06511
                 IF *jnestr* = *ojnestr* THEN                       06512
                    ptr _ jfnefln                                   06513
                 ELSE                                               06514
                    BEGIN                                           06515
                    IF jfnefln THEN                                 06516
                       BEGIN                                        06517
                       &fl _flntadr(jfnefln.stfile);                06518
                       fl.flnoclos _ FALSE;                         06519
                       END;                                         06520
                    *ojnestr* _ *jnestr*;                           06521
                    FIND SF(*jnestr*) ^tp1 SE(*jnestr*) ^tp2;
                                                                    06522
                    caddexp($tp1, $tp2, &da, &ptr);                 06523
                    jfnefln _ ptr;                                  06524
                    &fl _flntadr(jfnefln.stfile);                   06525
                    fl.flnoclos _ TRUE;                             06526
                    END;                                            06527
                 IF ptr.stpsid = origin THEN *st1* _ *st2*, "
                 .l"                                                06528
                 ELSE *st1* _ '!, *st2*, " .l";                     06529
                 FIND SF(*st1*) ^tp1 SE(*st1*) ^tp2;                06530
                 ON SIGNAL ELSE                                     06620
```

```
                        IF enftype THEN                                 06621
                           BEGIN                                        06616
                           ON SIGNAL ELSE;                              06622
                           enftype _ FALSE;  %skip file origin;
                           use user-profile only%                       06617
                           REPEAT CASE;                                 06618
                           END;                                         06619
                     caddexp($tp1, $tp2, &da, &ptr);                    06531
                     ON SIGNAL ELSE;                                    06615
                     END;                                               06532
                  ENDCASE;                                              06613
               END;                                                     06614
            IF ptr = endfil THEN *errwrk* _ *st2*;                      06533
            END;                                                        06534
         = ENDCHR:                                                      06535
            BEGIN                                                       06536
            slashflg _ savslh;                                          06537
            IF (slashflg) AND (NOT fnlsls) THEN                         06538
               BEGIN                                                    06539
               ccurcon( &ptr, $errwrk );                                06540
               typeas( $errwrk );                                       06541
               END;                                                     06542
            RETURN (da.davspec := savs1,                                06543
               da.davspc2 := savs2,                                     06544
                  da.dacacode := cacode,                                06545
                     da.dausqcod := useqgen,                            06546
                        rsrng);                                         06547
            END;                                                        06548
         ENDCASE                                                        06549
            GOTO caeerror;                                              06550
      IF ptr = endfil THEN GOTO caeerror;                               06551
      END;  % of loop %                                                 06552
   (caeerror): %error return%                                           06553
      slashflg _ savslh;                                                06554
      da.davspec _ savs1;                                               06555
      da.davspc2 _ savs2;                                               06556
      da.dacacode _ cacode;                                             06557
      da.dausqcod _ useqgen;                                            06558
      SIGNAL(gaderr, $errwrk);                                          06559
   END.

                                                                        06581
      %  %                                                             06561
```

```
%.....ADDRESS EXPRESSION SUPPORT ROUTINES...%                          02313
   (conspt)  % used for content and word searches %                    02969
      % determines type of search, number of elements to be found,
      and scope of the search.%                                        02970
      PROCEDURE(z1);                                                   02971
      LOCAL                                                            02972
         count, type, scope, tmp;                                      02973
      REF z1;                                                          02974
                                                                       02975
      count _ type _ scope _ tmp _ 0;                                  02976
      IF NOT FIND $(SP/TAB) '= THEN                                    02977
         RETURN( contnt, 1, -1);                                       02978
      FIND ^z1;                                                        06029
      CASE READC OF                                                    02979
         IN ['0, '9]:                                                  02980
            BEGIN                                                      02981
            FIND < CH >;                                               02982
            tmp _ gadnum();                                            02983
            FIND ^z1;                                                  06030
            REPEAT CASE;                                               02984
            END;                                                       02985
         = 'C, ='c:                                                    02986
            IF NOT type THEN                                           02987
               BEGIN                                                   02988
               FIND ^z1;                                               02989
               IF tmp THEN count _ tmp := 0 ELSE count _ 1;            02990
               type _ IF scope THEN cntls ELSE contnt;                 02991
               IF scope THEN RETURN( type, count, scope )              02992
               ELSE REPEAT CASE;                                       02993
               END                                                     02994
            ELSE                                                       02995
               BEGIN                                                   02996
               FIND z1;                                                02997
               RETURN( type, count, IF scope THEN scope ELSE -1 );
                                                                       02998
               END;                                                    02999
         = 'W, ='w:                                                    03000
            IF NOT type THEN                                           03001
               BEGIN                                                   03002
               FIND ^z1;                                               03003
               IF tmp THEN count _ tmp := 0 ELSE count _ 1;            03004
               type _ IF scope THEN wordls ELSE wordtyp;               03005
               IF scope THEN RETURN( type, count, scope )              03006
               ELSE REPEAT CASE;                                       03007
               END                                                     03008
            ELSE                                                       03009
               BEGIN                                                   03010
               FIND z1;                                                03011
               RETURN( type, count, IF scope THEN scope ELSE -1 );
                                                                       03012
               END;                                                    03013
         = 'S, ='s:                                                    03014
            IF NOT scope THEN                                          03015
               BEGIN                                                   03016
               FIND ^z1;                                               03017
```

```
            IF tmp THEN scope _ tmp := 0 ELSE scope _ 1;          03018
            CASE type OF                                          03019
               = FALSE:  REPEAT CASE 2;                           03020
               = wordtyp:                                         03021
                  RETURN( wordls, count, scope );                 03022
               = contnt:                                          03023
                  RETURN( contls, count, scope );                 03024
               ENDCASE err($"NLS System Error: CONSPT");          03025
            END                                                   03026
         ELSE                                                     03027
            BEGIN                                                 03028
            FIND z1;                                              03029
            RETURN(                                               03030
               IF type THEN type ELSE contnt,                     03031
                  IF count THEN count ELSE 1,                     03032
                     scope );                                     03033
            END;                                                  03034
      ENDCASE                                                     03035
         BEGIN                                                    03036
         FIND z1;                                                 03037
         RETURN(                                                  03038
            IF type THEN type ELSE contnt,                        03039
               IF count THEN count ELSE 1,                        03040
                  IF scope THEN scope ELSE -1 );                  03041
         END;                                                     03042
END.
                                                                  03043
%  %                                                             03044
```

```
(gadname) PROCEDURE(ptr, ast, type);                                    02341
   LOCAL TEXT POINTER tp1, tp2, tp3;                                    02342
   REF ptr, ast;                                                        02343
   FIND ^tp1;                                                           02344
   nmdr($tp1, $tp2, $tp3);                                              02345
   *ast* _ tp2 tp3;                                                     02346
   specreg(&ast, type, &ptr);                                           02347
   FIND tp3 >;                                                          02348
   RETURN;                                                              02349
   END.
                                                                        02350
   %  %                                                                 03537
```

```
   (gaddir) PROCEDURE(ptr,count);                                    02351
      REF ptr;                                                       02352
      IF count < 0 THEN                                              02353
         BEGIN                                                       02354
         FIND ptr < ;                                                02355
         RETURN ( -count, TRUE );                                    02356
         END;                                                        02357
      FIND ptr > ;                                                   02358
      RETURN ( count, FALSE );                                       02359
      END.

                                                                     03542
      %  %                                                          03543
```

```
   (getpr) PROCEDURE(proc, count, ptr);                                02360
      LOCAL lptr;                                                      05972
      REF proc, ptr;                                                   02361
      IF count < 0 THEN %do the inverse%                               02362
         BEGIN                                                         02363
         &proc _ CASE &proc OF                                         02364
            = $getsuc : $getprd;                                       02365
            = $getprd : $getsuc;                                       02366
            = $getsub : $getup;                                        02367
            = $getup  : $getsub;                                       02368
            = $gethed : $getail;                                       02369
            = $getail : $gethed;                                       02370
            = $getnxt : $getbck;                                       02371
            = $getbck : $getnxt;                                       02372
            ENDCASE &proc;                                             02373
         count _ -count;                                               02374
         END;                                                          02375
      UNTIL (count _ count-1) < 0 DO                                   02376
         BEGIN                                                         02377
         IF &proc = $getsuc AND getftl(ptr) THEN EXIT;                 02378
            % thus successor of statement with no successor is NOP %
                                                                       02379
         IF (lptr _ proc(ptr)) = endfil THEN EXIT;  %NOP%              05973
         ptr _ lptr;                                                   02380
         ptr[1] _ 1;                                                   02381
         END;                                                          02382
      RETURN;                                                          02383
      END.
                                                                       03545
      %  %                                                            03546
```

```
(gdlit2) PROCEDURE(ast, stopchar);                                  02384
   % append characters to ast from READC until encounter a
   stopchar or the end of input.%                                   02385
   LOCAL char;                                                      02386
   REF ast;                                                         02387
   LOOP                                                             02388
      BEGIN                                                         02389
      CASE char _ READC OF                                          02390
         =ENDCHR, =stopchar:                                        02391
            RETURN;                                                 02392
         ENDCASE                                                    02393
            *ast* _ *ast*, char;                                    02394
      END;                                                          02395
   END.

                                                                    03547
   %  %                                                            03548
```

```
(gadnum) PROCEDURE;                                                    02397
   % extract and evaluate number from current READC source.            %
                                                                       02398
   LOCAL count, char;                                                  02399
   IF (char _ READC) NOT IN [’0, ’9] THEN                              02400
      BEGIN                                                            06026
      IF char NOT= ENDCHR THEN FIND < CH >;                            06027
      RETURN(1);                                                       06025
      END;                                                             06028
   count _ char - ’0;                                                  02401
   WHILE (char _ READC) IN [’0, ’9] DO                                 02402
      count _ count*10 + char - ’0;                                    02403
   IF char NOT= ENDCHR THEN FIND < CH >;                               02404
   RETURN(count);                                                      02405
   END.

                                                                       03550
   %  %                                                               03551
```

```
%viewspec acceptor%
                                                                    06623
   (feedlt)  PROCEDURE(dpa, astrng);                                06624
      %Given the address of a display area, this routine changes
                                                                    06625
      its vspecs in accord with the specificaations in the          06626
      A-string passed it.  It passes the characters in the          06627
      A-string to <PRMSPC, SETLT>, except for content analyzer
                                                                    06628
      patterns.%                                                    06629
      %------------%                                               06630
      LOCAL count, length, char, vs1, vs2;                          06631
      LOCAL TEXT POINTER tp;                                        06632
      LOCAL STRING castng[250];                                     06633
      REF dpa, astrng;                                              06634
      length _ astrng.L;                                            06635
      count _ empty - 1;                                            06636
      vs1 _ dpa.davspec;                                            06637
      vs2 _ dpa.davspc2;                                            06638
      UNTIL (count _ count+1) > length DO                           06639
         IF (char _ *astrng*[count]) = '; THEN                      06640
            BEGIN                                                   06641
            *castng* _ NULL;                                        06642
            UNTIL (char _ *astrng*[count _ count + 1]) = '; DO
                                                                    06643
               BEGIN                                                06644
               *castng* _ *castng*, char;                           06645
               IF count >= length THEN EXIT LOOP1;                  06646
               END;                                                 06647
            *castng* _ *castng*, " ;;";                             06648
            FIND SF(*castng*) ^tp;                                  06649
            dpa.dacacode _ cpconan ($tp, &dpa);                     06650
            END                                                     06651
         ELSE vs1 _ setlt(char, vs1, vs2 : vs2);                    06652
      dpa.davspec _ vs1;                                            06653
      dpa.davspc2 _ vs2;                                            06654
      RETURN;                                                       06655
      END.                                                          06656
                                                                    06657
   (setlt)                                                          06658
      %This routine adjusts the viewspecs in accord with
      characters entered during view specification.  Saves the
      viewspec words on the stack savevspec for each character
      input.  When a BC is input the stack is popped.  The string
      being displayed in the name area is updated accordingly.%
                                                                    06659
      %---------------%                                            06660
      PROCEDURE(setchr, vs1, vs2);                                  06661
      LOCAL goodvs, settmp, vspec[2];                               06662
      vspec _ vs1;                                                  06663
      vspec[1] _ vs2;                                               06664
      goodvs _ TRUE;                                                06665
      CASE setchr OF                                                06666
         ='a: % 1_1-1 %                                             06667
            IF vspec.vsrlev THEN                                    06668
               BEGIN                                                06669
```

```
            IF vspec.vslev = 0 OR vspec.vslevd THEN                    06670
               BEGIN                                                   06671
               BUMP vspec.vslev;                                       06672
               vspec.vslevd _ TRUE;                                    06673
               END                                                     06674
            ELSE BUMP DOWN vspec.vslev;                                06675
            END                                                        06676
         ELSE IF vspec.vslev > 0 THEN BUMP DOWN vspec.vslev;
                                                                       06677
      ='b:  % l_l+1 %                                                  06678
         IF vspec.vslevd THEN                                          06679
            BEGIN                                                      06680
            BUMP DOWN vspec.vslev;                                     06681
            IF vspec.vslev = 0 THEN vspec.vslevd _ FALSE;              06682
            END                                                        06683
         ELSE IF vspec.vslev < 63 THEN BUMP vspec.vslev;               06684
      ='c:  % l_all %                                                  06685
         BEGIN                                                         06686
         vspec.vslev _ 63;                                             06687
         vspec.vsrlev _ vspec.vslevd _ FALSE;                          06688
         END;                                                          06689
      ='d:  % l_1 %                                                    06690
         BEGIN                                                         06691
         vspec.vslev _ 1;                                              06692
         vspec.vsrlev _ vspec.vslevd _ FALSE;                          06693
         END;                                                          06694
      ='e:  % l=rel %                                                  06695
         BEGIN                                                         06696
         IF vspec.vsrlev THEN %user is already in e state,
         reset previous e%                                             06697
            BEGIN                                                      06698
            (rstlev):                                                  06699
            vspec.vslev _ vspec.vsrlev;                                06700
            vspec.vsrlev _ FALSE;                                      06701
            vspec.vslevd _ FALSE;                                      06702
            END;                                                       06703
         vspec.vsrlev _ IF vspec.vslev THEN vspec.vslev ELSE
         TRUE;                                                         06704
         vspec.vslev _ FALSE;                                          06705
         vspec.vslevd _ FALSE;                                         06706
         END;                                                          06707
      ='g:  % branch only on %                                         06708
         BEGIN                                                         06709
         vspec.vsbrof _ TRUE;                                          06710
         vspec.vsplxf _ FALSE;                                         06711
         END;                                                          06712
      ='h:  % branch only / plex only off %                            06713
         BEGIN                                                         06714
         vspec.vsbrof _ FALSE;                                         06715
         vspec.vsplxf _ FALSE;                                         06716
         END;                                                          06717
      ='i:  % content analyzer success %                               06718
         BEGIN                                                         06719
         vspec.vscapf _ TRUE;                                          06720
         vspec.vscakf _ FALSE;                                         06721
         END;                                                          06722
```

```
='j:  % content analyzer off %                                  06723
   BEGIN                                                        06724
   vspec.vscapf _ FALSE;                                        06725
   vspec.vscakf _ FALSE;                                        06726
   END;                                                         06727
='k:  % content analyzer k flag %                               06728
   % only use content analyzer for first statement in
   sequence %                                                   06729
   BEGIN                                                        06730
   vspec.vscakf _ TRUE;                                         06731
   vspec.vscapf _ FALSE;                                        06732
   END;                                                         06733
='l:  % plex only on %                                          06734
   BEGIN                                                        06735
   vspec.vsplxf _ TRUE;                                         06736
   vspec.vsbrof _ FALSE;                                        06737
   END;                                                         06738
='m:  vspec.vsstnf _ TRUE;  % location numbers on %             06739
='n:  vspec .vsstnf _ FALSE;  % location numbers off %
                                                                06740
='o:  vspec.vsfrzf _ TRUE;  % frozen on %                       06741
='p:  vspec.vsfrzf _ FALSE;  % frozen off %                      06742
='q:  % t_t-1 %                                                 06743
   IF vspec.vstrnc > 1 THEN BUMP DOWN vspec.vstrnc;             06744
      %if this is changed so 0 is allowed, dafrmt must be
      fixed so it doesn't try to freeze every page in the
      file%                                                     06798
='r:  % t_t+1 %                                                 06745
   IF vspec.vstrnc < 63 THEN BUMP vspec.vstrnc;                 06746
='s:  % t_all %                                                 06747
   vspec.vstrnc _ 63;                                           06748
='t:  % t_1 %                                                   06749
   vspec.vstrnc _ 1;                                            06750
='u:  % display area formatter on %                             06751
   vspec.vsdaft _ TRUE;                                         06752
='v:  % display area formatter off %                            06753
   vspec.vsdaft _ FALSE;                                        06754
='w:  % l=t=all %                                               06755
   BEGIN                                                        06756
   vspec.vstrnc _ 63;                                           06757
   vspec.vslev _ 63;                                            06758
   vspec.vsrlev _ vspec.vslevd _ FALSE;                         06759
   END;                                                         06760
='x:  % l=t=1 %                                                 06761
   BEGIN                                                        06762
   vspec.vstrnc _ vspec.vslev _ 1;                              06763
   vspec.vsrlev _ vspec.vslevd _ FALSE;                         06764
   END;                                                         06765
='y:  vspec.vsblkf _ TRUE;  %  blank line on %                  06766
='z:  vspec.vsblkf _ FALSE;  % blank line off %                 06767
='A:  % indenting on, relative indenting off %                  06768
   BEGIN                                                        06769
   vspec.vsindf _ TRUE;                                         06770
   vspec.vsrind _ FALSE;                                        06771
   END;                                                         06772
='B:  % indenting off, relative indenting off %                 06773
```

```
            BEGIN                                                   06774
            vspec.vsindf _ FALSE;                                   06775
            vspec.vsrind _ FALSE;                                   06776
            END;                                                    06777
         ='C:  vspec.vsnamf _ TRUE;    % names on %                 06778
         ='D:  vspec.vsnamf _ FALSE;   % names off %                06779
         ='E:  vspec.vspagf _ TRUE;    % Paging on %                06780
         ='F:  vspec.vspagf _ FALSE;   % Paging off %               06781
         ='G:  vspec.vsstnr _ TRUE;    % statement numbers on right
         %                                                          06782
         ='H:  vspec.vsstnr _ FALSE;   % statement numbers on left
         %                                                          06783
         ='I:  vspec.vssidf _ TRUE;    % sid flag on %              06784
         ='J:  vspec.vssidf _ FALSE;    % sid flag off %            06785
         ='K:  vspec.vsidtf _ TRUE;    % initials, date, on %       06786
         ='L:  vspec.vsidtf _ FALSE;   % initials, date, off %
                                                                    06787
         ='O:  vspec.vsusqf _ TRUE;    % user sequence generator on
         %                                                          06788
         ='P:  vspec.vsusqf _ FALSE;   % user sequence generator
         off %                                                      06789
         ='Q:  % indenting and relative indenting on %              06790
            BEGIN                                                   06791
            vspec.vsrind _ TRUE;                                    06792
            vspec.vsindf _ TRUE;  % so Output Processors will be
            sure to show indenting on %                             06793
            END;                                                    06794
      ENDCASE goodvs _ FALSE;                                       06795
   RETURN(vspec, vspec[1], goodvs);                                 06796
   END.
                                                                    06797
%execute link%                                                      02406
   (nfstid)  %get t-ptr to new file and address expression%         02407
      PROCEDURE(fnmstng, stnstg, vspstg, da);                       02408
      %Given two strings, the first containing a file name, the
      second an address experssion, this routine will open the
      file, and return the corresponding stid and character count.
      Branch only viewspec is added to the viewspec string if this
      is a journal message.  DA is needed for address expression
      evaluation.%                                                  02409
      %-------------%                                               02410
      LOCAL vs1, vs2, cacode, useqgen, fno, jnlfg;                  02411
      LOCAL TEXT POINTER tptr, z1, z2;                              02412
      LOCAL STRING lkmkst[15], lkfnst[50];                          02413
      REF fnmstng, stnstg, vspstg, da;                              02414
      %Check to see if file is a journal file; if so, save off
      number%                                                       02415
         IF FIND SF(*fnmstng*) ('<['>]/) ^z1 4$D ^z2 THEN           02416
            BEGIN                                                   02417
            *lkfnst* _ 'J, z1 z2;                                   02418
            jnlfg _ TRUE;                                           02419
            END                                                     02420
         ELSE jnlfg _ FALSE;                                        02421
      tptr _ origin;                                                02422
      tptr.stfile _ fno _ cloafil(&fnmstng);                        02423
         %cloafil changes fnmstng%                                  02424
```

```
tptr[1] _ 1;                                                    06562
%Check to see if file is a journal message file%                02425
   IF jnlfg AND FIND SF(*fnmstng*) ('<[']/) "JRNL" THEN
                                                                02426
      BEGIN                                                     02427
      *stnstg* _ *lkfnst*;                                      02428
      END                                                       02429
   ELSE jnlfg _ FALSE;                                          02430
FIND SF(*stnstg*) ^z1 SE(*stnstg*) ^z2;                         02431
vs1 _ caddexp($z1, $z2, &da, $tptr : vs2, cacode, useggen);
                                                                02432
RETURN(tptr, tptr[1], vs1, vs2, cacode, useggen);               02433
END.
                                                                02434
%  %                                                           03552
```

```
%......statement name lookup......%
                                                                    02449
   (namelook)PROC(stid, astr);                                      02450
      %This routine accepts an stid and astring, an finds the
      named statemet in the file indicated by the stid, retruning
      the stid oof the statemet or endfil.  Uses non-sequential
      lookup.  Copies name astring into local string so a literal
      may be used in the call%                                      02451
      LOCAL TEXT POINTER z1;                                        02452
      LOCAL STRING namest[40];                                      02453
      REF astr;                                                     02454
      FIND SF(stid) ^z1;                                            02455
      *namest* _ *astr*;                                            02456
      lookup($z1, $namest, nametyp);                                02457
      RETURN(z1);                                                   02458
      END.
                                                                    02459
      %  %                                                         03554
```

```
(lookup)PROCEDURE(ptr, astrng, type);                                  02838
   %THis routine accepts a pointer, string, and type, and does
   a search through the file indicated by the pointer for the
   statemnt indicated by the string and type as follows:               02839
      type = nametyp                                                   02840
         searches the file for the first statement found with
         the indicated name.                                           02841
         Does a non-sequential search.                                 02842
         The string is modified.                                       02843
         Returns the stid of the statement as a result and in
         the pointer, or endfil on failure.                            02844
      type = nxtname                                                   02845
         Same as name, but starts from the place in the ring
         indicated by the stid in ptr.                                 02846
         Note that this also is a non-sequential search                02847
      type = seqname:                                                  02848
         Does a sequential search of the file for the next
         statemet (beginning with the one following the one
         indicated by ptr) with the indicated name.                    02849
         Returns stid in ptr, or endfil in ptr.                        02850
      type = braname                                                   02851
         Same as seqname, but search is restricted to branch
         headed by ptr.                                                02852
      type = contnt                                                    02853
         Does a sequential search of the file fo a statement
         with the content in string.                                   02854
         Search starts with the character following the one
         indicated by ptr                                              02855
         Returns same as seqname.                                      02856
      type = cont1s                                                    02857
         Same as content, but looks only in the current
         statement                                                     02858
      type = wordtyp                                                   02859
         Searches for word in string in a manner equivalent to
         content.                                                      02860
         Returns same as content.                                      02861
      type = word1s                                                    02862
         Same as wordtyp, but looks only in the current
         statement                                                     02863
      type = sid                                                       02864
         &astrng is assumed to be an sid, and te file is
         searched for a statement with a matching sid.                 02865
         Returns same as name.                                         02866
   %                                                                   02867
   %-------------------%                                              02868
   LOCAL nhash, count, sidval, p1scn, p2scn, stid;                     02869
   REF astrng, ptr;                                                    02870
   IF ptr = endfil THEN RETURN(endfil);                                02871
   CASE type OF                                                        02872
      = nametyp, =nxtname, =extname:                                   02873
         BEGIN                                                         02874
         astruc(&astrng);                                              02875
         nhash _ hash(&astrng);                                        02876
         stid _ IF type = nxtname THEN ptr ELSE 0;                     02877
         WHILE (stid _ lkupfast(ptr, stid, nhash, rnameh)) #
         endfil DO                                                     02878
```

```
            BEGIN                                                   02879
            CCPOS SF(stid);                                         02880
            xtrnam($lkupreg, $swork, -1, 0);                        02881
            IF *lkupreg* = *astrng* THEN                            02882
               RETURN(ptr _ stid);                                  02883
            END;                                                    02884
         RETURN(ptr _ endfil);                                      02885
         END;                                                       02886
      = seqname:                                                    02887
         BEGIN                                                      02888
         astruc(&astrng);                                           02889
         nhash _ hash(&astrng);                                     02890
         ptr[1] _ 1;                                                02891
         ptr _ getnxt(ptr);                                         02892
         END;                                                       02893
      = braname:                                                    02894
         BEGIN                                                      02895
         ptr[1] _ 1;                                                02896
         RETURN(ptr _ namingrp(ptr, ptr, &astrng, 1000));           02897
         END;                                                       02898
      = sid:                                                        02899
         RETURN(ptr _ lkupfast(ptr, 0, &astrng, rsid));             02900
      ENDCASE IF ptr[1]=empty THEN ptr[1]_1;                        02901
   UNTIL ptr = endfil DO                                            02902
      BEGIN                                                         02903
      IF inptrf THEN %have a rubout%                                02904
         BEGIN                                                      02905
         ptr _ endfil;                                              02906
         RETURN;                                                    02907
         END;                                                       02908
      CASE type OF                                                  02909
         =seqname:                                                  02910
            IF getnam(ptr) = nhash THEN                             02911
               BEGIN                                                02912
               CCPOS SF(ptr);                                       02913
               <UTILTY, xtrnam>($lkupreg, $swork, -1, 0);           02914
               IF <UTILTY, compas>($lkupreg, &astrng) THEN
                                                                    02915
               RETURN;                                              02916
               END;                                                 02917
         =contnt:                                                   02918
            IF FIND ptr > [*astrng*] ^ptr _ptr THEN RETURN;
                                                                    02919
         =cont1s: %search is limited to a single statement%
                                                                    02920
            BEGIN                                                   02921
            IF NOT FIND ptr > [*astrng*] ^ptr _ptr THEN ptr _
            endfil;                                                 02922
            RETURN;                                                 02923
            END;                                                    02924
         =wordtyp:                                                  02925
            BEGIN                                                   02926
            CCPOS ptr;                                              02927
            LOOP                                                    02928
               IF FIND [*astrng*] ^ptr _ptr < THEN                  02929
                  BEGIN                                             02930
```

```
                    count _ astrng.L;                                   02931
                    UNTIL (count _ count-1) < empty DO FIND CH;
                                                                        02932
                    IF FIND ^plscn -LD AND ptr > CH -LD THEN
                    RETURN;                                             02933
                    IF NOT FIND > plscn CH THEN EXIT;                   02934
                       % this can EXIT if astrng is empty and
                                                                        02935
                       have reached end of statement %                  02936
                    END                                                 02937
                 ELSE EXIT;                                             02938
              END;                                                      02939
           =wordls:                                                     02940
              BEGIN                                                     02941
              CCPOS ptr;                                                02942
              LOOP                                                      02943
                 IF FIND [*astrng*] ^ptr _ptr < THEN                    02944
                    BEGIN                                               02945
                    count _ astrng.L;                                   02946
                    UNTIL (count _ count-1) < empty DO FIND CH;
                                                                        02947
                    IF FIND ^plscn -LD AND ptr > CH -LD THEN
                    RETURN;                                             02948
                    IF NOT FIND > plscn CH THEN                         02949
                       % this can RETURN if astrng is empty and
                       have reached end of statement %                  02950
                       BEGIN                                            02951
                       ptr _ endfil;                                    02952
                       RETURN;                                          02953
                       END;                                             02954
                    END                                                 02955
                 ELSE                                                   02956
                    BEGIN                                               02957
                    ptr _ endfil;                                       02958
                    RETURN;                                             02959
                    END;                                                02960
              END;                                                      02961
           ENDCASE err($"NLS system error");                            02962
        ptr _ getnxt(ptr);                                              02963
        ptr[1] _ 1;                                                     02964
        END;                                                            02965
     RETURN                                                             02966
     END.
                                                                        02967
     % %                                                               02968
```

```
(specreg)                                                          02570
   %Spec a register.  This procedure converts a string or a
   statement identifier to a t-pointer.  The conversion
   algorithm depends on the first character in the register,
   and on the parameter passed as the second argument,  If a
   string is being matched, the routine expects as a third
   argument the stid at which the search should begin,      02571
      If the first character is a number, and a name is being
      speced, the register is assumed to contain a statement
      number, and FECHUX is used to convert the A-string to a
      STID.  Otherwise the register is assumed to contain
      either a word or a string to be matched.             02572
      If TYPE (the second argument) is                     02573
      1, the register is assumed to contain a name;        02574
      2, the register is assumed to contain a word;        02575
      3 or 6, the register is assumed to contain a string;
                                                           02576
      4, the register is assumed to contain a statement
      identifier;                                          02577
      In all of these cases LOOKUP is called to find the STID.%
                                                           02578
   %---------------%                                      02579
   PROCEDURE(astrng, type, ptr);                          02580
   REF astrng, ptr;                                       02581
   IF astrng.L = empty                                    02582
      THEN                                                02583
         BEGIN                                            02584
         ptr.stpsid _ origin;                             02585
         ptr[1] _ 1;                                      02586
         RETURN;                                          02587
         END;                                             02588
   IF type = nametyp OR type = seqname OR type = nxtname OR
   type = extname                                         02589
      THEN                                                02590
         BEGIN %number, sid or name%                      02591
         ptr[1] _ 1;                                      02592
         IF *astrng*[ 1 ] IN ['0, '9]                     02593
            THEN                                          02594
               BEGIN %number or SID%                      02595
               IF *astrng*[ 1 ] = '0 AND astrng.L > 1  % sid
               given %                                    02596
                  THEN                                    02597
                     BEGIN                                02598
                     *astrng* _ *astrng*[2 TO astrng.L];
                     %delete '0 %                         02599
                     lookup(&ptr, cvsno(&astrng), sid); %lookup
                     sid%                                 02600
                     RETURN;                              02601
                     END;                                 02602
               ptr _ fechux(&astrng, ptr.stfile);         02603
               RETURN;                                    02604
               END;                                       02605
         END;                                             02606
   lookup(&ptr, &astrng, type);                           02607
   RETURN;                                                02608
   END.
```

```
                                                                    02609
(nmdr) PROCEDURE (bug,ptr1,ptr2);                                   06594
   % statement name (and statement number) delimiter %              06595
   REF bug, ptr1, ptr2;                                             06596
   FIND bug >                                                       06597
      $(LD / '- /'' / '@ / '& / '{ / '} ) ^ptr2                     06598
      bug < $(LD / '- /'' / '@ / '& / '{ / '} ) ^ptr1;              06599
      % @ is alphabetic zero %                                      06600
   RETURN;                                                          06601
   END.                                                             06602
                                                                    06603
   %  %                                                            02610
```

```
(lkupfast)PROC(stid, start, value, field);                          02611
   LOCAL rnb, rnptr, rnt, pgindx, rngbkend, rngbkptr, fileno;
                                                                    02612
   REF rnptr, rngbkptr;                                             02613
   fileno _ stid.stfile;                                            02614
   rnt _ (rnb _ filehead[fileno]+$rngst-$filhed) + rngm;            02615
   &rnptr _ rnb+start.stblk-1;                                      02616
   IF start THEN                                                    02617
      BEGIN                                                         02618
      IF (start _ start.stwc + ringl) >= blksiz THEN  %Gone
      over to te next block%                                        02619
         BEGIN                                                      02620
         BUMP &rnptr;                                               02621
         start _ 0;                                                 02622
         END                                                        02623
      ELSE start _ start - fbhdl;                                   02624
      END;                                                          02625
   WHILE (&rnptr _ &rnptr+1) < rnt DO                               02626
      IF rnptr.rfexis THEN                                          02627
         BEGIN                                                      02628
         IF (pgindx _ rnptr.rfcore) = 0 THEN                        02629
            pgindx _ lodrfb(&rnptr-rnb+rngbas, rngtyp, fileno);
                                                                    02630
         rngbkend _ (&rngbkptr _ crpgad[pgindx] + fbhdl) +
         blksiz;                                                    02631
         IF start THEN &rngbkptr _ &rngbkptr + start := 0; %add
         in displacement if it's there%                             02632
         WHILE &rngbkptr < rngbkend DO                              02633
            IF rngbkptr.field = value THEN                          02634
               BEGIN                                                02635
               stid.stblk _ &rnptr-rnb;                             02636
               stid.stwc _ &rngbkptr-crpgad[pgindx];                02637
               RETURN(stid);                                        02638
               END                                                  02639
            ELSE &rngbkptr _ &rngbkptr + ringl;                     02640
         END;                                                       02641
   RETURN(endfil);                                                  02642
   END.
                                                                    03556
   % %                                                              03557
```

```
(namingrp) %lookup name in file starting at stid and
terminating when through with the branch headed by lbstid,
looking down levels levels.%                                    02644
   % This procedure finds the statement whose name is in
   namstr.  the search is restricted to that part of the file
   begining with STID through the branch headed by LBSTID, and
   within LEVELS levels below STID. %                           02645
   %--------------------%                                       02646
   PROC(stid, lbstid, namstr, levels);                          02647
   LOCAL rng, nhash, save;                                      02648
   LOCAL STRING locstr[100];                                    02649
   REF namstr, rng;                                             02650
   IF stid = endfil THEN RETURN(endfil);                        02651
   astruc(&namstr);                                             02652
   nhash _ hash(&namstr);                                       02653
   save _ rubabt := FALSE;                                      02654
   LOOP                                                         02655
      BEGIN                                                     02656
      IF getnmf(stid) AND getnam(stid) = nhash THEN             02657
         BEGIN                                                  02658
         CCPOS SF(stid);                                        02659
         xtrnam($locstr, $swork, -1, 0);                        02660
         IF *namstr* = *locstr* THEN                            02661
            BEGIN                                               02662
            rubabt _ save;                                      02663
            RETURN(stid);                                       02664
            END;                                                02665
         END;                                                   02666
      IF levels > 0 AND (stid := getsub(stid)) # stid THEN
                                                                02667
         BUMP DOWN levels                                       02668
      ELSE                                                      02669
         LOOP                                                   02670
            IF getftl(stid) THEN                                02671
               BEGIN                                            02672
               BUMP levels;                                     02673
               IF stid = lbstid OR inptrf THEN EXIT LOOP 2;
                                                                02674
               stid _ getsuc(stid); %GETSUC RETURNS THE UP IF
               TAIL%                                            02675
               END                                              02676
            ELSE                                                02677
               BEGIN                                            02678
               IF stid = lbstid OR inptrf OR stid.stpsid =
               origin THEN                                      02679
                  EXIT LOOP 2;                                  02680
               stid _ getsuc(stid);                             02681
               EXIT LOOP;                                       02682
               END;                                             02683
      END;                                                      02684
   rubabt _ save;                                               02685
   RETURN(endfil);                                              02686
   END.
                                                                02687
   %  %                                                         02688
```

FINISH 0179

AUX COD

```
< NLS, AUXCOD.NLS.49, >, 30-Mar-78 13:41 JDH ;;;;

FILE auxcod % L10 to <REL-NLS>Auxcod %% (L10,) (rel-nls,auxcod.rel,) %
                                                                    02
%.....declarations.....%
                                                                    03
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4, r5 = 5, p = 7, m = 10, s =
   9;                                                               04
   REF rawchr;                                                      05
   REF msgda, tda;                                                 0870
   SET JSYS=104B,                                                   07
      odtim = 220B, pmap=56B, haltf = 170B;                         08
%...Message Display...%                                            0677
   (dismes) %display a message to the user%
   PROCEDURE (type,astrng);                                        02253
      %This routine is called to display a message on the screen or tty.
      The address of an A-string is usually provided in Astrng.  Type
      contains an integer which determines the action taken as follows:
                                                                   02254
         Type = 0:                                                 02255
            will remove any message on the screen. An A-string need not
            be given in Astrng in this case.                       02256
            The global flag MSGRESET is set TRUE.                  02257
         Type = 1:                                                 02258
            the message will be displayed, and the routine will return
            (with the message still on the screen).                02259
            The global flag MSGRESET is set FALSE.                 02260
         Type = 2:                                                 02261
            causes the message to be displayed for a few seconds.
            dismes returns immediately however.                    02262
            The global flag MSGRESET is set FALSE  (TRUE when message is
            removed by msg fork pseudo-interrupt).                 02263
         Type >= 1000;                                             02264
            put the message up for type milliseconds.              02265
            The global flag MSGRESET is set FALSE  (TRUE when message is
            removed by msg fork pseudo-interrupt).                 02266
      %                                                            02267
      %--------------%                                            02268
      IF nldevice = offline THEN RETURN;                           02269
      IF nlmode = typewriter THEN                                  02270
         BEGIN                                                     02271
         msgreset _ TRUE;                                          02272
         IF type THEN                                              02273
            BEGIN                                                  02274
            crlf();                                                02275
            typeas(astrng);                                        02276
            END                                                    02277
         ELSE                                                      02278
            BEGIN                                                  02279
            crlf(); crlf();                                        02280
            END;                                                   02281
         RETURN;                                                   02282
         END;                                                      02283
      IF type = 0  AND msgreset THEN RETURN;                       02284
      IF nldevice = devlproc AND NOT tracking THEN                 02285
         track();                                                  02286
```

```
      %                                                              02287
      BEGIN let fork wait some more and then retry                   02288
      stoptimer();                                                   02289
      settimer(1000, $dismes, type, astrng);                         02290
      RETURN;                                                        02291
      END;                                                           02292
      %                                                              02293
   msgreset _ TRUE;                                                  02294
   IF type = 0 THEN                                                  02295
      BEGIN                                                          02296
      IF nldevice NOT= devlproc THEN                                 02297
         BEGIN                                                       02298
         crlf(); crlf();                                             02299
         END;                                                        02300
      timrset _ FALSE;                                               02470
      RETURN;                                                        02301
      END;                                                           02302
   %pause if timing currently being done%                            02303
      timerpause();                                                  02304
   CASE type OF                                                      02305
      =1: %put up and leave%                                         02306
         dismsg(astrng);                                             02307
      =2, >=1000: %put up for a few seconds%                         02308
         BEGIN                                                       02309
         settimer(IF type >= 1000 THEN type ELSE 3000, $dismes, 0,
         0);                                                         02310
         dismsg(astrng);                                             02311
         END;                                                        02312
      ENDCASE;                                                       02313
   RETURN END.                                                       02314
                                                                     02315
(timerpause) PROCEDURE;                                              02334
   LOCAL i;                                                          02335
   FOR i _ 0 UP UNTIL >= 3 DO                                        02336
      IF timrset THEN !disms(2000)                                   02337
      ELSE EXIT LOOP;                                                02338
   timrset _ FALSE;                                                  02396
   RETURN;                                                           02339
   END.                                                              02340
(dismsg) %..display the message for DISMES..%
PROCEDURE(astrng);                                                   01717
   LOCAL TEXT POINTER tp1;                                           01718
   LOCAL STRING tstr[100];                                           01719
   REF astrng;                                                       01720
   msgreset _ FALSE;                                                 01721
   IF (nldevice = devlproc) AND (tenex >= 13200) THEN                01722
      BEGIN  % TEMP KLUDGE UNTIL BBN FIXES %                         01723
      crlf();                                                        01724
      typeas($"");                                 01725
      FIND SF(*astrng*) ^tp1;                                        01726
      LOOP                                                           01727
         IF FIND tp1 > [LF] ^tp1 THEN                                01728
            BEGIN                                                    01729
            *tstr* _ SF(*astrng*) tp1;                               01730
            *astrng* _ tp1 SE(*astrng*);                             01731
            typeas($tstr);                                           01732
```

```
            typeas($"");                                    01733
            tp1[1] _ 1;                                                 01734
            END                                                         01735
         ELSE EXIT LOOP;                                                01736
      typeas(&astrng);                                                  01737
      IF astrng.L < (msgda.daright-msgda.daleft)/msgda.dahinc AND NOT
      (FIND SF(*astrng*) [EOL/CR LF]) THEN                              01738
         BEGIN                                                          01739
         crlf();                                                        01740
         typeas($"");                                   01741
         END;                                                           01742
      END                                                               01743
   ELSE                                                                 01744
      BEGIN                                                             01745
      crlf();                                                           01746
      typeas(&astrng);                                                  01747
      IF astrng.L < (msgda.daright-msgda.daleft)/msgda.dahinc AND NOT
      (FIND SF(*astrng*) [EOL/CR LF]) THEN                              01748
         crlf();                                                        01749
      END;                                                              01750
   RETURN;                                                              01751
   END.
                                                                        01752
(settimer)  %set the system timer -- after MILLISECONDS milliseconds,
PROC will be called with ARG1 thru ARG4%
PROCEDURE (milliseconds, proc, arg1, arg2, arg3, arg4);                 02316
   LOCAL frkac0, frkac1;                                                02317
   stoptimer();                                                         02320
   timrset _ TRUE;                                                      02321
   timrproc _ proc;                                                     02322
   timra1 _ arg1;                                                       02323
   timra2 _ arg2;                                                       02324
   timra3 _ arg3;                                                       02325
   timra4 _ arg4;                                                       02326
   frkac1 _ MIN(10000, milliseconds); %milliseconds to wait%            02327
   !sfacs(msgfrk, $frkac0); %set timer fork acs%                        02328
   !stork(msgfrk, 120B); %start the timer fork%                         02329
   !rfork(); %thaw fork, r1 already set%                                02330
   RETURN;                                                              02331
   END.                                                                 02332
                                                                        02333
(stoptimer)  %stop the system timer%
PROCEDURE;                                                              01183
   %should be able to simply halt the fork, but 10X calls it an error
   to halt a fork which is already halted, STUPID STUPID%               01184
   !ffork(msgfrk); %freeze the fork%                                    01186
   !rfsts(msgfrk); %read status%                                        01187
   IF NOT r1 .A 2B6 THEN                                                01188
      !hfork(msgfrk); %halt the fork%                                   01189
   timrset _ FALSE;                                                     01191
   RETURN;                                                              01192
   END.                                                                 01193
                                                                        01194
(msgpsi) %pseudo interrupt from the timer fork%
PROCEDURE;                                                              01195
   svac1 _ r1;                                                          01196
```

```
   r1 _ $svacs;                                                    01197
   !BLT r1, svacse;                                                01198
   s _ s + 40000040B;                                              01199
   timrset _ FALSE;                                                01200
   %call the designated procedure%                                 01201
      [timrproc](timra1, timra2, timra3, timra4);                  01202
   r1.LH _ $svacs;                                                 01203
   r1.RH _ 0;                                                      01204
   !BLT r1, 17B;                                                   01205
   r1 _ svac1;                                                     01206
   !JSYS debrk;                                                    01207
   END.

                                                                   01208
(typeas) PROCEDURE (astrng); %Type a-string on tty%                 0876
   %-----------------%                                             0877
   REF astrng;                                                     0878
   IF NOT astrng.L THEN RETURN;                                    0879
   !sout(101B, chbmty + &astrng, -astrng.L);                       0883
   RETURN;                                                         0884
   END.

                                                                   0885
(typech) PROCEDURE(char); %type (translated) character on tty%     01095
   LOCAL tchar, spclcharstr;                                       01096
   REF spclcharstr;                                                01097
   %-----------------%                                             01098
   IF (r1 _ trnslo[char]) NOT= nullch THEN !pbout;                 01099
   RETURN                                                          01100
   END.

                                                                   01101
(crlf) PROCEDURE; %type a carriage return-line feed%               0871
   %-----------------%                                             0872
   !pbout(trnslo[CR]);                                             0873
   !pbout(trnslo[LF]);                                            01858
   RETURN;                                                         0874
   END.

                                                                   0875
(cnvcrlfteol) PROCEDURE(string); %convert string with CRLF's to
EOL's%                                                             02220
   LOCAL srcptr, desptr, hptr, cnt, i;                             02221
   REF string;                                                     02222
   srcptr _ desptr _ chbptr(0) + &string;                          02223
   cnt _ string.L;                                                 02224
   i _ 0;                                                          02225
   WHILE i < cnt DO                                                02226
      BEGIN                                                        02227
      IF (^desptr _ ^srcptr) = CR THEN                             02228
         BEGIN                                                     02229
         hptr _ srcptr;                                            02230
         IF ^srcptr = LF THEN                                      02231
            BEGIN                                                  02232
            BUMP DOWN cnt;                                         02233
            .desptr _ EOL;                                         02234
            END                                                    02235
         ELSE srcptr _ hptr;                                       02236
         END;                                                      02237
      BUMP i;                                                      02238
```

```
      END;                                                          02239
   string.L _ cnt;                                                  02240
   ^despTr _ 0;                                                     02243
   RETURN;                                                          02241
   END.                                                             02242
(cnveoltcrlf)PROCEDURE(t1, string); %convert string with EOL's to
CRLF's%                                                             02244
   %start at pointer t1, go to string end.  put converted stuff in
   string%                                                          02245
   LOCAL TEXTPOINTER t4;                                            02414
   REF t1, string;                                                  02415
   *string* _ NULL;                                                 02416
   FIND SE(t1) ^t4 > ;                                              02417
   cetcapp(&t1, $t4, &string);                                      02418
   RETURN;                                                          02419
   END.                                                             02420
(cetcapp)PROCEDURE(t1, t4, string); %eol-to-crlf convert string
segment t1 t4, append to string%                                    02405
   %start at pointer t1, go to string end.  put converted stuff in
   string%                                                          02406
   LOCAL TEXT POINTER t2, t3;                                       02407
   REF t1, string, t4;                                              02408
   WHILE (FIND BETWEEN t1 t4 (^t2 [EOL] ^t1 ^t3 _ t3)) DO *string* _
   *string*, t2 t3, CR, LF;                                         02410
   *string* _ *string*, t1 t4;                                      02411
   RETURN;                                                          02412
   END.                                                             02413
%.....error, abort, and termination routines.....%                   010
(goroot) PROCEDURE;  %Goto root%                                    0671
   SIGNAL(statesig, 0);                                             0672
   END.                                                             0673
(gps) PROCEDURE;  %GOTO STATE routine%                              0674
   SIGNAL(statesig, 0);                                             0675
   END.
                                                                    0676
(nlsrst) PROCEDURE;  %RESET NLS to rcover from castastrophic
problems%                                                           0886
   supervisor();                                                    0887
   halt();                                                          0889
   END.
                                                                    0888
(rerror) PROCEDURE;                                                  092
   LOCAL STRING temp[50];                                            093
   dismes(2, $"Fatal error"); %leave for 1 second%                   094
   ddgtsymbol($temp, [m .A 18M] .A 18M -1);                          095
   typeas($"Crash at PROCEDURE: ");                                  096
   typeas($temp);                                                    097
   (errhlt):                                                         098
      !JSYS 147B; %Reset jsys-- cannot use symbol because of
      conflict%                                                      099
      !JSYS haltf; GOTO errhlt;                                     0100
      %In case someone is foolish enough to try to contnue%         0101
   END.                                                             0102
                                                                    0103
(tenwheel) PROCEDURE(type, fileno, filename); %check for wheel%     02425
   IF NOT enwheel() THEN RETURN(FALSE);                             02427
```

```
   RETURN(otflerrec(type, fileno, filename));                          02468
   END.                                                                02469
(otflerrec) PROCEDURE(type, fileno, filename); %err record, file name
format%                                                                02467
   LOCAL recloc, dwptr, fl, dwc, i, er;                                02426
   REF filename, dwptr, fl, er;                                        02441
   recloc _ getblk(40, $dspblk); %get buffer%                          02428
   &er _ recloc+bhl;                                                   02442
   CASE type OF                                                        02433
      =3:                                                              02434
         BEGIN                                                         02437
         errfill(&er, type, 0);                                        02429
         RETURN(FALSE);                                                02465
         END;                                                          02438
      ENDCASE                                                          02436
         BEGIN                                                         02439
         er.ercode _ type;                                             02443
         !gjinf();                                                     02444
         er.eruser _ r1;                                               02445
         er.erjobn _ r3;                                               02446
         &dwptr _ &er + 5;                                             02447
         dwptr.M _ 200;                                                02462
         IF fileno THEN                                                02456
            BEGIN                                                      02457
            &fl _ flntadr(fileno);                                     02463
            *dwptr* _ *[fl.flastr]*;                                   02464
            END                                                        02458
         ELSE                                                          02459
            BEGIN                                                      02460
            *dwptr* _ *filename*;                                      02449
            END;                                                       02461
         er.erdlen _ 0;                                                02450
         FOR i _ 1 UP UNTIL > 5 DO                                     02451
            BEGIN                                                      02453
            IF outerrec(&er) THEN EXIT LOOP;                           02452
            !dismS(100);                                               02455
            END;                                                       02454
         END;                                                          02440
   RETURN(TRUE);                                                       02430
   END.                                                                02431
                                                                       02432
(iodaterr)PROC;                                                         0104
   %Come here on an IO data Error (channel 11)%                         0105
   [levtab] _ $ioderr;                                                  0106
   !JSYS debrk;                                                         0107
   (ioderr):                                                            0108
      SIGNAL(-6, $"I/O Data Error");                                    0109
      %We don't know what file the error was on, so for now just type
      a nasty mesage and signal.  deferr will rerror%                   0110
   END.                                                                 0111
(thwrfp) PROCEDURE; %thaw random file pages%                            0112
   LOCAL end, ct;                                                       0113
   REF ct;                                                              0114
   end _ (&ct _ $corpst + 1) + rfpmax;                                  0115
   DO IF ct.ctpnum # 0 THEN ct.ctfroz _ 0                               0116
   UNTIL (&ct _ &ct + 1) = end;                                         0117
```

```
   RETURN;                                                          0118
   END.                                                             0119
                                                                    0120
(thwfil) PROC(filno); %thaw file pages of file%                     0121
   LOCAL end, ct;                                                   0122
   REF ct;                                                          0123
   end _ (&ct _ $corpst + 1) + rfpmax;                              0124
   DO IF ct.ctfile = filno AND ct.ctpnum # 0 THEN ct.ctfroz _ 0     0125
   UNTIL (&ct _ &ct + 1) = end;                                     0126
   RETURN;                                                          0127
   END.                                                             0128
                                                                    0129
(pause) PROCEDURE(tim);                                             0130
   % pause for specified number of msec %                           0131
   r1 _ tim;                                                        0132
   !JSYS 167B; %disms%                                              0133
   RETURN END.
                                                                    0134
(halt) PROCEDURE; %terminate NLS%                                   0135
   %close work station (if NLS), set continue, and issue            0136
   terminate jsys%                                                  0137
   %--------------%                                                 0138
   IF nlmode NOT= typewriter THEN shutdis();                        0140
   ctlquit();                                                       0141
   continue _ TRUE;                                                 0142
   closeall();                                                      0143
   IF msjfn THEN sysclose(msjfn:=0);                               01887
   !JSYS haltf;                                                     0145
   GOTO rentr;   % if a person forgets and does a CONTINUE rather
   than a RENTER after an Execute Quit, then this will do the RENTER
   for him %                                                        0146
   END.                                                             0147
                                                                    0148
(shutdis) PROCEDURE; %shut down display before leave NLS%          01753
   LOCAL STRING send[10];                                           01754
   LOCAL da, ls, end, rtend;                                        01755
   REF ls, da;                                                      01756
   CASE nldevice OF                                                 01757
      = devlproc:                                                   01758
         BEGIN                                                      01759
         lprset();                                                  01760
         cscreen();  %just to make sure%                            01761
         vspsav _ vspsav[1] _ 0; %viewspecs will be refreshed upon
         reentry%                                                   01762
         litdahandle _ ttysim _ 0;                                  01763
         clrall(0, FALSE);                                          01764
         END;                                                       01765
      ENDCASE                                                       01766
         BEGIN                                                      01767
         IF defttysim THEN                                          01768
            BEGIN                                                   01769
            vspsav _ vspsav[1] _ 0; %viewspecs will be refreshed upon
            reentry%                                                01770
            &da _ $dpyarea;                                         01771
            end _ $dpyend;                                          01772
            UNTIL &da >= end DO                                     01773
```

```
            BEGIN                                                   01774
            IF da.daexis THEN                                       01775
               BEGIN                                                01776
               dealocda(&da);                                       01777
               &ls _ da.dalsrt;                                     01778
               rtend _ da.dalsz * lsrtl + da.dalsrt - 1;            01779
                UNTIL &ls >= rtend DO                               01780
                  BEGIN                                             01781
                  ls.rtlsid _ 0;                                    01782
                  &ls _ &ls + lsrtl;                                01783
                  END;                                              01784
               END;                                                 01785
            &da _ &da + dal;                                        01786
            END;                                                    01787
         &da _ $nlsdas;                                             01788
         end _ $nlsdae;                                             01789
         UNTIL &da >= end DO                                        01790
            BEGIN                                                   01791
            IF da.daexis THEN                                       01792
               BEGIN                                                01793
               dealocda(&da);                                       01794
               &ls _ da.dalsrt;                                     01795
               rtend _ da.dalsrt;                                   01796
                UNTIL &ls >= rtend DO                               01797
                  BEGIN                                             01798
                  ls.rtlsid _ 0;                                    01799
                  &ls _ &ls + lsrtl;                                01800
                  END;                                              01801
               END;                                                 01802
            &da _ &da + dal;                                        01803
            END;                                                    01804
         litdahandle _ ttysim _ 0;                                  01805
         CASE nldevice OF                                           01806
            =imlac0, =imlac1:                                       01807
               BEGIN                                                01808
               *send* _ begmsg, 2+remfudge, echda, 45B;             01809
               !sout(dspjfn, chbmty+$send, -send.L);                01810
               *send* _ begmsg, 1+remfudge, remtsn;                 01811
               !sout(dspjfn, chbmty+$send, -send.L);                01812
               END;                                                 01813
            ENDCASE                                                 01814
               err($"No tasker");                                   01856
         END;                                                       01817
      END;                                                          01818
   IF tenex < 13200 THEN                                            01819
      !sbcim(0) %turn big char input mode off%                      01820
   ELSE IF nldevice = devlproc THEN                                 01821
      BEGIN                                                         01822
      !bout( dspjfn, lpesc );                                       01823
      !bout( dspjfn, lpnocoor );                                    01824
      END;                                                          01825
   RETURN END.
                                                                    01826
(gofork) PROCEDURE(prcnam, injfn, outjfn, ntiW); %goto lower fork%
                                                                    0890
   %run a lower level fork with the specified process, input jfn and
```

```
output jfn, leave ntiw psi on%                                          0891
LOCAL                                                                   0892
   savnm,  %saved name from tenex%                                     01315
   tmode, %terminal mode word%                                          0893
   pjfn, %jfn for process%                                              0894
   pfork,  %fork handle for process%                                    0895
   tiw;  % terminal interupt word %                                     0896
% close measurement file if open%                                     01888
   IF msjfn THEN sysclose(msjfn:=0);                                   01889
% get and remeber tenex subsystem name %                              01316
   !getnm();                                                           01317
   savnm _ r1;                                                         01318
%create fork%                                                           0897
   r1 _ 2B11; %pass capabilites%                                        0898
   IF NOT SKIP !JSYS 152B %cfork% THEN                                  0899
      err($"Can't create fork");                                        0900
   pfork _ r1;                                                          0901
%enable capabilities%                                                  0902
   r1 _ pfork;                                                          0903
   r2 _ r3 _ -1; %enable all%                                           0904
   !JSYS 151B; %epcap%                                                  0905
%set primary input if given jfn%                                       0906
   IF injfn # -1 OR outjfn # -1 THEN BEGIN                              0907
      r1 _ pfork;                                                       0908
      r2.LH _ injfn; r2.RH _ outjfn;                                    0909
      !JSYS 207B; %spjfn%                                               0910
      END;                                                              0911
%get process into the new fork%                                        0912
   IF NOT (pjfn _ lgetjfn($subdir,prcnam,$savext,gtjprf,$lit))
   THEN err($"Can't get jfn for process");                              0913
   r1.LH _ pfork;  r1.RH _ pjfn;  !JSYS get;                            0914
%shut down NLS display%                                                0915
   IF nlmode = fulldisplay THEN shutdis();                              0916
%save terminal mode word%                                              0917
   r1 _ 100B;  !JSYS rfmod;  tmode _ r2;                                0918
%reset terminal mode%                                                  0919
   r1 _ 100B;  r2 _ 20510175520B;  !JSYS sfmod;                         0920
% save tiw this fork %                                                 0921
   !rtiw( 400000B );                                                   0922
   tiw _ r2;                                                           0923
% set tiw to ntiw for this fork %                                      0924
   !stiw( 400000B, ntiw);                                              0925
%start fork using entry vector%                                        0926
   r1 _ pfork;  r2 _ 0;  !JSYS 201B; %sfrkv%                           0927
%wait for process to terminate%                                        0928
   r1 _ pfork;  !JSYS 163B; %wfork%                                    0929
%kill fork%                                                            0930
   r1 _ pfork;  !JSYS 153B;  %kfork%                                   0931
%release process jfn%                                                  0932
   reljfn(pjfn);                                                       0933
% reset tiw for this fork %                                           0934
   !stiw( 400000B, tiw);                                               0935
%restore subsystem name%                                               0936
   !setnm( savnm );                                                   01319
%restore terminal mode word%                                           0939
   r1 _ 100B; r2 _ tmode; !JSYS sfmod;                                 0940
```

```
   %restore NLS display%                                                0941
      ckdvln(); %see if lp line number changed%                        01857
      continue _ TRUE;                                                 0942
      <FINTNLS, initdis>();                                            0943
      continue _ FALSE;                                                0944
   %continue measurement if it was going%                              01890
      meastart();                                                      01891
   %return%                                                            0945
      RETURN END.
                                                                       0946
%...Pseudo-interrupt routines...%                                      0780
   (rubout) PROCEDURE; %rubout interupt routine%                       02027
      %---------------------%                                         02028
                                                                       02029
      %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%               02030
                                                                       02031
      %---------------------%                                         02032
      %save registers; IMPORTANT:  do this first%                      02035
         svac1 _ r1;                                                   02036
         r1 _ $svacs;                                                  02037
         !BLT r1, svacse;                                              02038
      IF msjfn THEN  %monitoring commands%                             02033
         msrecord(mstop, 0, 0);                                        02034
      IF libflg THEN killib();                                         02039
      IF rubabt:= FALSE THEN [levtab] _ $rabort                        02040
      ELSE [rubmrk] _ TRUE;                                            02041
      %clear input buffer%                                             02042
         !cfibf(777777B); %clear input buffer%                         02043
         IF NOT rubnocob THEN !cfobf();    %also clear output buffer%
                                                                       02059
         buffn _ buffs; % empty input buffer %                         02044
      %restore registers%                                              02045
      !HRLZI r1, svacs;                                                02046
      !BLT r1, 17B;                                                    02047
      r1 _ svac1;                                                      02048
      !debrk;                                                          02049
      (rabort): %come here for real aborts%                            02050
         rubsig();                                                     02051
      END.
                                                                       02052
   (rubsig) PROC;                                                      0806
      %simply calls signal for rubout%                                 0807
      SIGNAL(statesig);                                                0808
      END.
                                                                       0809
   (stopline) PROCEDURE; % ^S interrupt routine%                       0810
      %---------------------%                                         0811
                                                                       0812
      %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%               0813
                                                                       0814
      %---------------------%                                         0815
      %set flag%                                                       0818
      !AOS inpstp;                                                     01320
      !JSYS debrk;                                                     0822
      END.
                                                                       0823
```

```
(trapc)  PROCEDURE; %no-op control c%                                   0824
   !SOSLE ccignore;                                                     0825
   !JSYS debrk;                                                         0826
   GOTO [savchntab[2]];                                                 0827
   END.
                                                                        0828
(trapo)  PROCEDURE; %no-op control o%                                   0829
   !SOSLE ccignore;                                                     0830
   !JSYS debrk;                                                         0831
   GOTO [savchntab[3]];                                                 0832
   END.
                                                                        0833
(traps)  PROCEDURE; %no-op control s%                                   0834
   !SOSLE ccignore;                                                     0835
   !JSYS debrk;                                                         0836
   GOTO [savchntab[1]];                                                 0837
   END.
                                                                        0838
(joctlc)PROC;                                                           0967
   %Come here on a control c%                                           0968
   !JSYS 141B; %cis--clear intterupt system (can't us set because of
   duplicate symbol)%                                                   0969
   jclosfl();                                                           0970
   jctlcres();                                                          0971
   r1 _ -1; %exec%                                                      0972
   r2 _ 2B11; %^C channel%                                              0973
   !JSYS iic; %cause interrupt%                                         0974
   r1 _ 2000; !JSYS disms; %for TENEX timing problem%                   0975
   dismes(2,$"Journal Process Aborted by ^C");                          0976
   nlsrst();                                                            0977
   END.
                                                                        0978
(jctlcres)PROC;                                                         0979
   IF NOT jsavaccess THEN disabaccess(0, jrnlaccess);                   0980
   %connect back to original directory%                                 0981
      conjdir(FALSE);                                                   0982
   %De-activate ^C channel%                                             0983
      r1 _ 4B5;                                                         0984
      !JSYS rcm; %read channel mask--134%                               0985
      IF r1 .A 1B11 THEN                                                0986
         BEGIN %de-activate it%                                         0987
         r1 _ 4B5;                                                      0988
         r2 _ 1B11;                                                     0989
         !JSYS dic;                                                     0990
         r1 _ 3;                                                        0991
         !JSYS dti;                                                     0992
         END;                                                           0993
   RETURN;                                                              0994
   END.                                                                 0995
                                                                        0996
(brkconnection)  PROCEDURE; %break display screen connection%           0839
   %set program counter to actual routine%                              0840
   !MOVEM r1,svac1;                                                     0841
   !MOVEI r1,brkclabel;                                                 0842
   !MOVEM r1,@levtab;                                                   0843
   !MOVE r1,svac1;                                                      0844
```

```
      !JSYS debrk;                                                    0845
      (brkclabel):                                                    0846
         rstconnection();                                             0847
         GOTO STATE;                                                  0848
      END.
                                                                      0849
   (rstconnection)  PROCEDURE; %break display screen connection%     0850
      %break links%                                                   0853
         IF NOT SKIP !tlink(6B11 .V 777777B, 777777B) THEN NULL;      0854
         %break adviz%                                                0855
         IF NOT SKIP !adviz(4B11) THEN NULL;                          0856
      %shut down NLS display%                                         0857
         shutdis();                                                   0858
      IF ldspjfn THEN                                                01322
         BEGIN                                                       01323
         IF NOT SKIP !closf(ldspjfn) THEN NULL;                      01324
         reljfn(ldspjfn);                                            01325
         END;                                                        01326
      linkcnsl _ ldspjfn _ 0;                                         0859
      %restore NLS display%                                           0860
         %restore old format%                                         0861
            IF savnldevice NOT= nldevice THEN setdev(savnldevice);    0862
         continue _ TRUE;                                             0863
         <INTNLS, initdis>();                                         0864
         continue _ FALSE;                                            0865
         alldsp();                                                    0866
      chntab[1] _ savchntab[1];                                       0867
      RETURN;                                                         0868
      END.
                                                                      0869
%...call stack underflow routine...%                                  0267
   (uflow) PROCEDURE; %general stack underflow routine%               0268
      s _ m _ -$gstksz;                                               0269
      !HRL m,m; !HRL s,s;                                             0270
      !HRRI s,gstack; !HRRI m,gstack;                                 0271
      state _ $gstack + 2;                                            0272
      state[1] _ $supervisor;                                         0273
      state[2] _ $gstack;                                             0274
      state[3] _ $edit;                                               0275
      dismes(2, $"Call stack underflow -- report circumstances to ARC
      staff");                                                        0276
      supervisor();                                                   0277
      halt();                                                         0278
      END.
                                                                      0279
%.....help routines.....%                                             0280
   (changcom)PROC(string, retparm);                                   0281
      %This procedure accepts a string as a parameter, and types out a
      message (dismes) saying tht th command has been changed, and that
      the user should consult Folklore for details%                   0282
      %Returns in the same manner as help%                            0283
      LOCAL count;                                                    0284
      LOCAL STRING msgstring[250];                                    0285
      REF string;                                                     0286
      IF (count _ string.L) + 50 >msgstring.M THEN count _ 200;       0287
      *msgstring* _ CR, LF, *string*[1 TO count], CR, LF, "Command
```

```
      Changed--see (documentation, folklore,)";                          0288
      dismes(2, $msgstring);                                             0289
      IF retparm = -1 THEN RETURN;                                       0290
      IF retparm = -2 THEN SIGNAL(statesig);                             0291
      GOTO STATE;                                                        0292
      END.                                                               0293
%.....marker code.....%                                                  0294
   (delmkn) PROCEDURE (fileno, name);                                    0295
      %delete the marker whose name is passed from the specified         0296
      file%                                                              0297
      %------------------%                                              0298
      LOCAL marker, end, flhd, mkrcount, stid, aring;                    0299
      REF marker, mkrcount, aring;                                       0300
      % make sure we have a locked file %                               01309
         stid _ origin;                                                  01310
         stid.stfile _ fileno;                                           01311
         lodent( stid, rngtyp : &aring );                                01312
         aring.rsub _ aring.rsub;                                        01313
      flhd _ filhdr(fileno) - $filhed;                                   0301
         %subtract $filhed here instead of throughout procedure%         0302
      &marker _ flhd + $mkrtb;                                           0303
      end _ &marker + [&mkrcount _ flhd + $mkrtbl]*mkrl;                 0304
      LOOP                                                               0305
         BEGIN                                                           0306
         IF &marker >= end THEN RETURN; %no such marker%                 0307
         IF marker.mkname = name THEN EXIT;                              0308
         &marker _ &marker + mkrl;                                       0309
         END;                                                            0310
      mvbfbf(&marker+mkrl, &marker, end-&marker-mkrl);                   0311
      mkrcount _ mkrcount - 1;                                           0312
      RETURN;                                                            0313
      END.                                                               0314
                                                                         0315
   (delmkr) PROCEDURE (tptrs);                                           0316
      %delete the markers in T-string specified by arg%                  0317
      LOCAL marker, end, flhd, mkrcount;                                 0318
      REF marker, mkrcount;                                              0319
      flhd _ filhdr([tptrs].stfile) - $filhed;                           0320
      &marker _ flhd + $mkrtb;                                           0321
      end _ &marker + [&mkrcount _ flhd + $mkrtbl]*mkrl;                 0322
      UNTIL &marker >= end DO                                            0323
         IF marker.mkpsid = [tptrs].stpsid AND                           0324
         marker.mkccnt IN [[tptrs+1],[tptrs+3]) THEN                     0325
            BEGIN                                                        0326
            mvbfbf( &marker+mkrl, &marker, end-&marker-mkrl);            0327
            mkrcount _ mkrcount - 1;                                     0328
            end _ end - mkrl;                                            0329
            END                                                          0330
         ELSE &marker _ &marker + mkrl;                                  0331
      RETURN;                                                            0332
      END.                                                               0333
                                                                         0334
   (insmkr) PROCEDURE (bug, name);                                       0335
      %insert marker%                                                    0336
      LOCAL marker, end, flhd, mkrcount, newmkr, mkrmaxlen;              0337
      REF marker, mkrcount, mkrmaxlen;                                   0338
```

```
    flhd _ filhdr([bug].stfile) - $filhed;                                  0339
    &marker _ flhd + $mkrtb;                                                0340
    &mkrmaxlen _ flhd + $mkrtxn;                                            0670
    end _ &marker + [&mkrcount _ flhd + $mkrtbl]*mkrl;                      0341
    newmkr _ TRUE;                                                          0342
    UNTIL &marker >= end DO                                                 0343
       BEGIN                                                                0344
       IF marker.mkname = name THEN                                         0345
          BEGIN                                                             0346
          newmkr _ FALSE;                                                   0347
          EXIT;                                                             0348
          END;                                                              0349
       &marker _ &marker + mkrl;                                            0350
       END;                                                                 0351
    IF newmkr THEN                                                          0352
       BEGIN                                                                0353
       IF mkrcount * mkrl >= mkrmaxlen THEN                                 0354
          err($"Marker table too long -- new marker not added");            0355
       BUMP mkrcount;                                                       0356
       marker.mkname _ name;                                                0357
       END;                                                                 0358
    marker.mkpsid _ [bug].stpsid;                                           0359
    marker.mkccnt _ [bug+1];                                                0360
    RETURN;                                                                 0361
    END.                                                                    0362
                                                                            0363
(seemkr) PROCEDURE (fileno, astr);                                          01260
    LOCAL marker, end, flhd, mkrcount, word, stid, count, char;             01261
    LOCAL TEXT POINTER tptr;                                                01262
    LOCAL STRING locstr[40], lstr2[40];                                     01263
    REF marker, mkrcount, astr;                                             01264
    flhd _ filhdr(fileno) - $filhed;                                        01265
    &marker _ flhd + $mkrtb;                                                01266
    end _ &marker + [flhd + $mkrtbl]*mkrl;                                  01267
    stid _ 0;                                                               01268
    stid.stfile _ fileno;                                                   01269
    *astr* _ NULL;                                                          01270
    UNTIL &marker >= end DO                                                 01271
       BEGIN                                                                01272
       *locstr* _ "    ";                                                   01273
       count _ 0;                                                           01274
       word _ marker.mkname;                                                01275
       UNTIL (count _ count + 1) = 6 DO                                     01276
          BEGIN                                                             01277
          char _ CASE count OF                                              01278
             =1: word.chr0;                                                 01279
             =2: word.chr1;                                                 01280
             =3: word.chr2;                                                 01281
             =4: word.chr3;                                                 01282
             ENDCASE word.chr4;                                             01283
          IF char # 0 THEN *locstr* _ *locstr*, char                        01284
          ELSE *locstr* _ *locstr*, SP;                                     01285
          END;                                                              01286
       stid.stpsid _ marker.mkpsid;                                         01287
       % get current location to astr %                                     01288
          tptr _ stid; tptr[1] _ marker.mkccnt;                             01289
```

```
          ccurloc( $tptr, $lstr2 );                                    01290
       *astr* _ *astr*, *locstr*, *lstr2*, CR, LF;                     01291
       &marker _ &marker + mkrl;                                       01292
       END;                                                            01293
    RETURN;                                                            01294
    END.                                                               01295
                                                                       01296
%.....time/date routine, ident.....%                                    0455
(getdat) %put date and time in astring passed%                          0456
    PROCEDURE(astng);                                                  0457
    dtfrmt(-1, astng); %format current date%                           0458
    RETURN;                                                            0459
    END.                                                               0460
                                                                       0461
(dttrmt) %put date and time in canonical form in astring passed%        0462
    PROCEDURE(tim, astng);                                             0463
    REF astng;                                                         0465
    datfrmt( tim, 201B6, &astng);                                      0669
    RETURN;                                                            0473
    END.                                                               0474
                                                                       0475
(daofrmt) %put date only in canonical form in astring passed%          01859
    PROCEDURE(tim, astng);                                             01860
    REF astng;                                                         01861
    datfrmt( tim, 601B6, &astng);                                      01862
    RETURN;                                                            01863
    END.                                                               01864
                                                                       01865
(datfrmt) %put date and time in string passed%                         02361
    %tim is internal TENEX format, however TOPS20 format is handled
    since some these got into NLS files during the early days of ISID.
    The time is put in astng assuming "lcltimzon" as time zone
    (typically 8 for Pacific time).  This leaves the possibility of
    having "lcltimzon" a useroption.%                                  02395
    PROCEDURE (tim, frmt, astng);                                      02362
    LOCAL bytptr, count, savr5;                                        02363
    REF astng;                                                         02364
    IF tim = -1 THEN tim _ gtadcall();                                 02365
    IF tim.RH >= 250600B THEN %assume it's an old t20 format still
    hanging around%                                                    02366
       BEGIN                                                           02367
       !HRRZ r3,tim;                                                   02369
       !IMULI r3,250600B;                                              02370
       !ADDI r3,400000B; %rounding%                                    02371
       !HLRM r3,tim;                                                   02372
       END;                                                            02373
    %now have it in TENEX internal format%                             02374
    IF tops20flag THEN %convert to tops20 internal format%             02375
       BEGIN                                                           02376
       !HRRZ r3,tim;                                                   02377
       !IMULI r3,604271B;                                              02378
       !ADDI r3,100000B;                                               02379
       !LSH r3,2; %day fraction (*1B6) in r3.LH%                       02380
       !HLRM r3,tim;                                                   02381
       END;                                                            02382
    bytptr _ chbptr(astng.L) + &astng;                                 02383
```

```
   !odcnv(0, tim, 0, 1B11+lcltimzon*1B6);                                02384
   r1 _ bytptr;                                                          02385
   savr5 _ r5; % because this register is used by compilers, it must
   be saved and restored %                                               02421
   r5 _ frmt;                                                            02386
   !odtnc();                                                             02387
   r5 _ savr5;                                                           02422
   count _ slngth(bytptr, r1);                                           02388
   IF astng.L + count > astng.M THEN err($"NLS internal error, string
   too long");                                                           02389
   astng.L _ astng.L + count;                                            02390
   RETURN;                                                               02391
   END.                                                                  02392
(gtadcall)PROCEDURE;                                                     02341
   %get internal date and time in TENEX (right half is seconds since
   GMT midnight) format%                                                 02393
   !gtad();                                                              02342
   IF NOT tops20flag THEN RETURN(r1);                                    02343
   !HRRZ r3,r1; %fraction of day * 1B6 in r3.RH%                         02347
   !IMULI r3,250600B; %seconds in a day%                                 02348
   !ADDI r3,400000B; %rounding, seconds since midnight GMT in r3.LH%
                                                                         02349
   !HLR r1,r3; %put back in r1.RH%                                       02350
   RETURN (r1);                                                          02351
   END.                                                                  02352
(idtimcall)PROCEDURE(str, flags);                                        02471
   %call idtim for string "str" and flags "flags", returns true if
   successful; second argument is TENEX internal format%                 02472
   %smashes r3%                                                          02482
   REF str;                                                              02473
   IF NOT SKIP !idtim(chbmty+&str, flags) THEN RETURN(FALSE);            02474
   IF tops20flag THEN %convert to TENEX format%                          02475
      BEGIN                                                              02476
      !HRRZ r3,r2;                                                       02478
      !IMULI r3,250600B; %#seconds in day%                               02479
      !ADDI r3,400000B; %round%                                          02480
      !HLR r2,r3 ; %#seconds since gmt midnight in r2.RH%                02481
      END;                                                               02477
   RETURN(TRUE, r2);                                                     02483
   END.                                                                  02484
                                                                         02485
(totentad)PROCEDURE; %convert tops20 interal time to tenex%              02397
   %expects tops20 format time in r1, returns tenex format time in
   r1, smashes r3%                                                       02404
   !HRRZ r3,r1; %fraction of day * 1B6 in r3.RH%                         02400
   !IMULI r3,250600B; %seconds in a day%                                 02401
   !ADDI r3,400000B; %rounding, seconds since midnight GMT in r3.LH%
                                                                         02402
   !HLR r1,r3; %put back in r1.RH%                                       02403
   RETURN;                                                               02398
   END.                                                                  02399
(gmtfset)PROCEDURE;                                                      02353
   !PUSH s,r1; %preserve r1%                                             02354
   !odcnv(0,-1,0,0); %find local time zone%                              02355
   gmtflag _ IF NOT SKIP !TLNE r4,77B THEN -1 ELSE 1;                    02356
      %1 if zone is GMT, -1 if not%                                      02357
```

```
   !POP s,r1;                                                          02358
   RETURN;                                                             02359
   END.                                                                02360
(idfrmt) %put ident into astring passed%                               01866
   PROCEDURE(stid, astng);                                             01867
   LOCAL sdbadr, word, bytptr, count, char, stdb;                      01873
   REF sdbadr, astng;                                                  01874
   % eventually want to change this to work for any property type %
                                                                       01875
      IF NOT lodprop( stid, txttyp :&sdbadr, stdb) THEN                01876
         err($" No text block with this node");                        01877
   % initials %                                                        01878
      word _ getint(stdb);                                             01879
      bytptr _ stbptr(empty) + $word;                                  01880
      count _ 0;                                                       01881
      UNTIL (count _ count + 1) > 4 DO                                 01882
         IF (char _ ^bytptr) = 0 THEN EXIT LOOP                        01883
         ELSE *astng* _ *astng*, char;                                 01884
   RETURN;                                                             01870
   END.                                                                01871
                                                                       01872
%.....routines to support name delimiter commands.....%                 0476
(nmdlsset) %***% PROCEDURE  % set name delimiters for a statement %
                                                                       01827
   (stid, dlleft, dlrght);                                             01828
   LOCAL rnl, sdb;                                                     01829
   LOCAL STRING str[100];                                              01830
   REF rnl, sdb;                                                       01831
   lodent(stid, rngtyp : &rnl);                                        01832
   IF NOT lodprop( stid, txttyp : &sdb) THEN                           01833
      err($"No text block associated with this node.");                01834
   sdb.slnmdl _ dlleft;                                                01835
   sdb.srnmdl _ dlrght;                                                01836
   CCPOS SF(stid);                                                     01837
   *str* _ NULL;                                                       01838
   xtrnam ($str, $swork, dlleft, dlrght);                              01839
   IF str.L = empty THEN                                               01840
      BEGIN                                                            01841
      sdb.sname _ 1;                                                   01842
      rnl.rnameh _ 0;                                                  01843
      rnl.rnamef _ FALSE;                                              01844
      END                                                              01845
   ELSE                                                                01846
      BEGIN                                                            01847
      sdb.sname _ swork1;                                              01848
      astruc($str);                                                    01849
      rnl.rnameh _ hash($str);                                         01850
      rnl.rnamef _ TRUE;                                               01851
      END;                                                             01852
   RETURN;                                                             01853
   END.                                                                01854
                                                                       01855
(xnmdlset) PROCEDURE  % Xecute set name deilimiters for a group %
                                                                        0518
   (stid1, stid2, dlleft, dlrght, da);                                  0519
   LOCAL sw, stid;                                                      0520
```

```
    REF da, sw;                                                          0521
    &sw _ openseq (stid1, stid2, da.davspec, da.davspc2, da.dausgcod,
    da.dacacode);                                                        0522
    UNTIL (stid _ seqgen(&sw)) = endfil DO                               0523
       nmdlsset (stid, dlleft, dlrght);                                  0524
    closeseq (&sw);                                                      0525
    RETURN;                                                              0526
    END.                                                                 0527
                                                                         0528
%.....file status routine.....%                                          0550
  (fstatus) PROCEDURE(fileno, astrng, type);                             0551
    LOCAL fstfdb[25B], drn, nt, numst, fl, cnt, usd, tot, st, percent,
    i, stid;                                                             0552
    LOCAL STRING dname[25];                                              0553
    REF st, fl, astrng;                                                  0554
    &fl _ flntadr(fileno);                                               0555
    %file name%                                                          0556
       filnam( fileno, &astrng );                                        0947
       *astrng* _ *astrng*, CR, LF;                                      0557
    %private file?%                                                     01298
       IF rdprvsts (fileno) = $psprivate THEN                           01299
          BEGIN                                                         01300
          *astrng* _ *astrng*, "Private File";                          01301
          stid _ 0;                                                     01302
          stid.stpsid _ origin;                                         01303
          stid.stfile _ fileno;                                         01304
          IF NOT getfacc (stid, 0, 0, 0) THEN                           01305
             *astrng* _ *astrng*, " (but with no Access List)";         01306
          *astrng* _ *astrng*, CR, LF;                                  01307
          END;                                                          01308
    % get fdb for Modifications and Size %                             01332
       !gtfdb( fl.florig, 25B6, $fstfdb);                                0560
          %read entire FDB%                                              0561
    IF type .A 2 THEN %being modified?%                                  0558
       BEGIN                                                             0559
       drn _ fstfdb[24B].lkdirn;                                         0562
       nt _ fstfdb[24B].lkinit;                                          0563
       IF drn # 0 OR nt # 0 THEN                                         0564
          <NLS, FILMNP, lockid>(&astrng, drn, nt)                        0565
       ELSE *astrng* _ *astrng*, "File not being modified";              0566
       *astrng* _ *astrng*, CR, LF;                                      0567
       %browse?%                                                         0568
          IF fl.flbrws THEN *astrng* _ *astrng*, "In temporary
          modifications mode", CR, LF;                                   0569
       END;                                                              0570
    IF type .A 4 THEN %directory default for links%                      0571
       BEGIN                                                             0572
       IF NOT <IOEXEC, gdftdir>(fileno, $dname) THEN *dname*  _ "not
       on this system";                                                  0573
       *astrng* _ *astrng*, "Default directory for links is ",           0574
          *dname*, CR, LF;                                               0575
       END;                                                              0576
    IF type = 7 THEN %version creation time%                             0577
       BEGIN                                                             0578
       %this version%                                                    0583
          *astrng* _ *astrng*, "Creation date of this version: ";        0584
```

```
      r1 _ fstfdb[13B];                                              02423
      IF tops20flag THEN totentad(); %convert r1 to tenext time
      format%                                                        02424
      dtfrmt(r1, &astrng);                                           0585
      *astrng* _ *astrng*, CR, LF;                                   0586
   END;                                                              0587
IF type .A 1 THEN %size%                                             0588
   BEGIN                                                             0589
   %number of statements%                                            0590
      numst _ <NLS, VERIFY, crng>(TRUE, fileno);                     0591
      *astrng* _ *astrng*, STRING(numst), " statements in file",
      CR, LF;                                                        0592
   %structure pages%                                                 0593
      &st _ filehead[fileno] + $rngst - $filhed;                     0594
      cnt _ 0;                                                       0595
      usd _ tot _ blksiz;                                            0596
      FOR i _ 0 UP UNTIL = rngm DO                                   0597
         BEGIN                                                       0598
         IF st.rfexis THEN                                           0599
            BEGIN                                                    0600
            BUMP cnt;                                                0601
            usd _ usd + st.rfused;                                   0602
            tot _ tot + blksiz;                                      0603
            END;                                                     0604
         BUMP &st;                                                   0605
         END;                                                        0606
      *astrng* _ *astrng*,                                           0607
         "Structure pages = ", STRING(cnt), "/, STRING(rngm), CR,
         LF;                                                         0608
   %data pages%                                                      0609
      &st _ filehead[fileno] + $dtbst - $filhed;                     0610
      cnt _ 0;                                                       0611
      FOR i _ 0 UP UNTIL = dtbm DO                                   0612
         BEGIN                                                       0613
         IF st.rfexis THEN                                           0614
            BEGIN                                                    0615
            BUMP cnt;                                                0616
            usd _ usd + st.rfused;                                   0617
            tot _ tot + blksiz;                                      0618
            END;                                                     0619
         BUMP &st;                                                   0620
         END;                                                        0621
      *astrng* _ *astrng*,                                           0622
         "Data pages = ", STRING(cnt), "/, STRING(dtbm), CR, LF;
                                                                     0623
   %total pages%                                                     0624
      *astrng* _ *astrng*,                                           0625
         "Total pages in file = ", STRING(fstfdb[11B].RH), CR, LF;
                                                                     0626
   %used words%                                                      0627
      *astrng* _ *astrng*,                                           0628
         STRING(usd), " words used out of ", STRING(tot),            0629
         " words in file (=", STRING(percent _
         (usd*100+tot/2)/tot), "%)";                                 0630
   %percent used too low?%                                           0631
      IF percent < 70 AND fstfdb[11B].RH > 3 THEN                    0632
```

```
            *astrng* _ *astrng*, CR, LF, CR, LF,                    0633
               "Try an Update File Compact to improve % used";      0634
      END;                                                          0635
   RETURN END.
                                                                    0636
%.....group allocation.....%                                        0637
   %
   This code provides for "deleteing" jobs from the group allocation
   data page for the nls commands "Execute Logout"
   %                                                                0638
   (lockpage) PROCEDURE(core); %lock the data page%                 0639
      LOCAL                                                         0640
         count;                                                     0641
      FOR count _ 20 DOWN UNTIL = 0 DO                              0642
         BEGIN                                                      0643
         IF SKIP !AOSE @core THEN                                   0644
            BEGIN                                                   0645
            [core+1] _ gtadcall();      %set time of locking%       0647
            RETURN(TRUE);                                           0648
            END;                                                    0649
         !disms (1000); %wait a sec%                                0650
         END;                                                       0651
      RETURN(FALSE);                                                0652
      END.                                                          0653
%.....measurement.....%                                            01892
   (meastart)PROCEDURE;                                            01893
      %start or restart measurement if conditions satisfied%       01894
      LOCAL jobno, blkadr, indx, wrd, ltime;                       01895
      LOCAL STRING msflstr[80], tempsr[120];                       01896
      REF wrd;                                                     01897
      IF msjfn THEN RETURN; %already going%                        01898
      !gjinf();                                                    01899
      jobno _ r3;                                                  01900
      IF NOT blkadr _ flgpage( :indx) THEN RETURN; %couldn't get flag
      file%                                                        01901
      IF jobno > [blkadr+ffjobmax] THEN GOTO meas1; %jobno to big for
      flag file tables%                                            01902
      &wrd _ blkadr + ffacctime; %pointer to last flagfile acc. time%
                                                                   01903
      ltime _ !time(); %time since system came up, probably
      milliseconds%                                                01904
      !EXCH r1,@wrd; %old time in r1%                              01905
      IF r1 - ltime > 60*r2 THEN flgminit(blkadr); %initialize flag
      tables if this is first NLS entry after system startup%      01906
         %tough luck if there is a sixty second delay from the !time to
         the !EXCH%                                                01907
      !runtm(-5); r4 _ r2; %console time in r3, divisor in r4%     01908
      !time(); %up time in r1, divisor in r2%                      01909
      !time _ (r1/r2 - r3/r4); %up time at login in seconds%       01910
      &wrd _ blkadr + ffjobltab + jobno;                           01911
      IF (wrd.msfltime - ltime IN [-5,5] AND wrd.msfmflag) OR
      strtmeasure(blkadr, jobno, &wrd, ltime) THEN                 01912
         BEGIN %open measure file%                                 01913
         msflname($msflstr, wrd, blkadr); %make up file name (file
         should already exist)%                                    01914
         IF (msjfn _ sgtjfn(getgtjflg(read, 0, oldvrsn), $msflstr,
```

```
      $tempsr)) AND NOT SKIP !openf(msjfn, 70000020000B) THEN
      reljfn(msjfn:=0);                                              01915
      IF msjfn THEN msrtcop(wrd, blkadr); %initialize msrtab%        01916
      %initialize "last" cells and job share globals%                02056
         !jobtm(); mlastcpu _ r1;                                    02057
         !gtrpi(400000B); mlasttraps _ r1; mlastfaults _ r2;         02058
         !sysgt(getsbn(IF lhostn=utilhost THEN $"NJBGRP" ELSE
         $"NAPROC"));                                                02195
            %"NAPROC" doesn't exist at o-1, and can't use "NJBGRP" in
            the same way at other places because it's floating at o-1
            and integer elsewhere%                                   02211
         mnaproc _ r2;                                               02196
         !sysgt(getsbn($"DSHARE"));                                  02197
         mdshare _ r2;                                               02198
         !gjinf();                                                   02199
         jobno _ r3;                                                 02200
         !sysgt(getsbn($"PIEGRP"));                                  02201
         IF r2 AND mnaproc AND mdshare THEN                          02202
            BEGIN                                                    02203
            r2.LH _ jobno;                                           02204
            IF SKIP !getab(r2) THEN mnaproc.LH _ mdshare.LH _ r1
                                                                     02205
            ELSE mnaproc _ 0;                                        02206
            END                                                      02207
         ELSE mnaproc _ 0;                                           02208
         msumrap _ msumrcnt _ 0;                                     02209
      END;                                                           01917
   %map out flag file and release core page%                         01918
      (meas1):                                                       01919
         !pmap (-1, 4B11 .V (blkadr/1000B), 0);                      01920
         frzblk (indx, -1);                                          01921
         sysclose(flagjfn:=0);                                       01922
   RETURN; END.                                                      01923
(msrtcop)PROCEDURE(wrd, ba);                                         01924
   %if flag file has a record flag table for this meas. type, copy it
   to msrtab%                                                        01925
   LOCAL loc;                                                        01926
   IF (loc _ [ba + wrd.msfmtyp + ffrftab]) THEN mvbfbf(loc+ba,
   $msrtab, msrtl);                                                  01927
   RETURN;                                                           01928
   END.                                                              01929
(strtmeasure) PROCEDURE (ba, jobno, wrd, ltime);                     01930
   %decide if this login should do measurement and if so, create the
   output file%                                                      01931
   LOCAL fcnt, mtype;                                                01932
   REF fcnt, wrd;                                                    01933
   FOR mtype _ 0 UP UNTIL >= msfmtmax DO                             01934
      IF mstypcheck(mtype, ba, &wrd, ltime) THEN                     01935
         BEGIN                                                       01936
         &fcnt _ ba + fffcnt;                                        01937
         !AOS r1,@fcnt;                                              01938
         wrd.msffno _ r1;                                            01939
         wrd.msfmtyp _ mtype;                                        01940
         wrd.msfmflag _ 1;                                           01941
         wrd.msfltime _ ltime;                                       01942
         IF NOT mscrfile (&wrd, ba) THEN EXIT LOOP;                  01943
```

```
        RETURN(TRUE);                                                   01944
        END;                                                            01945
   %all tests failed, mark as "not to be measured"%                     01946
   wrd.msfmflag _ 0;                                                    01947
   wrd.msfltime _ ltime;                                                01948
   RETURN(FALSE);                                                       01949
   END.                                                                 01950
(mstypcheck) PROCEDURE (mtype, ba, wrd, ltime);                         01951
   %see if this login should be measured with measure type mtype%
                                                                        01952
   LOCAL gcnt, gmod;                                                    01953
   REF wrd, gcnt;                                                       01954
   IF NOT [ba+fftftab+mtype] THEN RETURN(FALSE);                        01955
   CASE mtype OF                                                        01956
      =mstu1, =mstu2: %unconditional start if wrd has been set%         01957
         IF wrd.msfmtyp=mtype AND ltime < wrd.msfltime THEN
         RETURN(TRUE);                                                  01958
      IN [mstg1, mstg2]: %periodic sampling%                            01959
         BEGIN                                                          01960
         gmod _ [ba + ffglmod + mtype - mstg1];                         01961
         &gcnt _ ba + ffglcnt + mtype - mstg1;                          01962
         !AOS r1,@gcnt;                                                 01963
         !IDIV r1,gmod;                                                 01964
         IF r2=0 THEN RETURN(TRUE);                                     01965
         END;                                                           01966
      ENDCASE;                                                          01967
   RETURN(FALSE);                                                       01968
   END.                                                                 01969
(msflname)PROCEDURE(fstr, lwrd, blkadr);                                01970
   %get measurement filename according to flagfie entry at wrd%         01971
   LOCAL prind, ct, kv, prstr;                                          01972
   LOCAL STRING ctstr[10];                                              01973
   REF prstr, fstr;                                                     01974
   prind _ lwrd.msfmtyp;                                                01975
   IF NOT (&prstr _ [blkadr + ffprtab + prind] ) THEN                   01976
      *fstr* _ '<, *userstr*, ">M", *initsr*, ".MSR"                    01977
   ELSE                                                                 01978
      BEGIN                                                             01979
      &prstr _ &prstr + blkadr;                                         01980
      DIV lwrd.msffno / 512, kv, ct;                                    01981
      *ctstr* _ STRING(ct);                                             01982
      WHILE ctstr.L < 3 DO *ctstr* _ '0, *ctstr*;                       01983
      *fstr* _ *prstr*, *ctstr*, ".MSR";                                01984
      END;                                                              01985
   RETURN; END.                                                         01986
(mscrfile)PROCEDURE(wrd, ba);                                           01987
   %create measurement file according to flagfile entry at wrd%         01988
   LOCAL cnt;                                                           01989
   LOCAL STRING mfstr[80], tempsr[120];                                 01990
   REF wrd;                                                             01991
   msflname($mfstr, wrd, ba);                                           01992
   IF (msjfn _ sgtjfn(getgtjflg(write, 0, dfltvrs), $mfstr, $tempsr))
   AND NOT SKIP !openf(msjfn, 70000020000B) THEN reljfn(msjfn:=0);
                                                                        01993
   IF NOT msjfn THEN RETURN(FALSE);                                     01994
   %put in any header stuff%                                            01995
```

```
      !bout(msjfn, 101B);  %type 1%                                    02190
      mswdstore(gtadcall() .A 770000B, 1, 18);  %current time to
      approx hour%                                                     02191
      !time();                                                         02192
      mswdstore(r2, 3, 18);  %clock time divisor%                      02193
      msrtcop(wrd, ba); %be sure msrtab set%                           01996
      FOR cnt _ 0 UP UNTIL >=msrtl DO mswdstore(msrtab[cnt], 6, 0);
                                                                       01997
      IF nlmode = fulldisplay THEN !bout(msjfn, $ctld) %DNLS%          02212
      ELSE !bout(msjfn, $ctlt);  %TNLS%                                02213
      !bout(msjfn, $ctla);  %end of header%                            02210
   %close file%                                                        01998
      IF NOT sysclose(msjfn:=0, $tempsr) THEN RETURN(FALSE);           02000
   RETURN(TRUE);                                                       02001
   END.                                                                02002
(mswdstore)PROCEDURE(wd, chrs, skbits);                                02003
   %put out chrs characters from wd to  measurement file%              02004
      %skip most significant skbits bits%                              02005
      %characters are in [100B,177B]%                                  02006
   LOCAL cnt;                                                          02026
   r1 _ msjfn;                                                         02007
   r3 _ wd;                                                            02008
   !LSH r3,@skbits;                                                    02009
   FOR cnt _ chrs DOWN UNTIL <= 0 DO                                   02010
      BEGIN                                                            02011
      r2 _ 1;                                                          02012
      !LSHC r2,6;                                                      02013
      !bout;                                                           02014
      END;                                                             02015
   RETURN; END.                                                        02016
(flgminit)PROCEDURE(ba);                                               02017
   %reset flag file job table at ba%                                   02018
   LOCAL table, jobmax, cnt;                                           02019
   REF table;                                                          02020
   &table _ ba + ffjobltab;                                            02021
   jobmax _ [ba + ffjobmax];                                           02022
   FOR cnt _ 0 UP UNTIL >jobmax DO table[cnt] _ 0;                     02023
   RETURN;                                                             02024
   END.                                                                02025
(msrecord) PROCEDURE (rectype, str, size);                             02112
   LOCAL                                                               02113
      codeword, codebit, code,       %for getting record code byte%
                                                                       02114
      i,                             %FOR loop index%                 02115
      ptraps, pfaults, ptime,        %paging info%                    02116
      truncerr;                      %truncation error occurred%
                                                                       02117
   REF str;                                                            02118
                                                                       02119
   %initialize to say no truncation errors have occurred%              02120
      truncerr _ FALSE;                                                02121
   %Get the record code byte for this record type (used to decide
   what info to put out for the record).  There's a different one for
   each record type; each is in the format:
     Bit 8  record this type (left-most bit)
     Bit 7  clock time
```

```
      Bit 6  string information
      Bit 5  count information
      Bit 4  CPU time
      Bit 3  pager traps
      Bit 2  page faults
      Bit 1  page routine time
      Bit 0  unused (right-most bit)
   %
      DIV rectype/msrmod, codeword, codebit;                          02122
      msrrlp _ msrr0[codebit] %a record def'n% + codeword;            02123
      %construct a record pointer to the correct byte%
      code _ msrtab.msrrlp;  %get that byte from the set%             02124
   %Check whether to put out anything for this record type%           02125
      IF code .A 400B %the high-order bit% THEN                       02126
         !bout(msjfn, $ctlr)  %cntl-R to start a record%              02127
      ELSE RETURN;  %don't put out anything%                          02128
   %Put out a code letter for the record type%                        02129
      !bout(msjfn, rectype+100B);                                     02130
   %Put out a clock time?%                                            02131
      IF code .A 200B THEN                                            02132
         BEGIN                                                        02133
         !time();  %get time in ms in R1%                             02134
         !LSH r1,-4;                                                  02135
         mswdstore(r1, 3, 18);  %put out rightmost 18 bits as 3       02189
         characters with ASCII code > 100%
         END;                                                         02136
   %Put out (command word or load average) string?%                   02137
      IF code .A 100B THEN                                            02138
         BEGIN                                                        02139
         !sout(msjfn, chbmty+&str, -MIN(str.L, msstrlng), 0);         02140
         IF str.L < msstrlng THEN  %make string long enough%          02141
            FOR i _ str.L UP 1 UNTIL = msstrlng DO                    02142
               !bout(msjfn, SP);                                      02143
         END;                                                         02144
   %Put out the size?%                                                02145
      IF code .A 40B THEN                                             02146
         mswdstore(size, 2, 24);                                      02147
   %Put out the CPU time?%                                            02148
      IF code .A 20B THEN                                             02149
         BEGIN                                                        02150
         !jobtm();  %to get job CPU info%                             02151
         IF r1 - mlastcpu > 1B6 THEN truncerr _ 20B;  %note that      02152
         truncation will lose some information%
         mlastcpu _ r1;                                               02153
         mswdstore(mlastcpu, 3, 18);                                  02154
         END;                                                         02155
   %Put out paging info?%                                             02156
      IF code .A 16B THEN                                             02157
         BEGIN  %get and save page info%                              02158
         !gtrpi(485);                                                 02159
         ptraps _ r1;                                                 02160
         pfaults _ r2;                                                02161
         ptime _ r3;                                                  02162
         IF code .A 10B THEN                                          02163
            BEGIN  %page trap info%                                   02164
            IF ptraps - mlasttraps > 1B4 THEN                         02165
                                                                      02166
```

```
            truncerr _ truncerr .V 10B;                                  02167
         mlasttraps _ ptraps;                                            02168
         mswdstore(ptraps, 2, 24);                                       02169
         END;                                                            02170
      IF code .A 4B THEN                                                 02171
         BEGIN  %page fault info%                                        02172
         IF pfaults - mlastfaults > 1B4 THEN                             02173
            truncerr _ truncerr .V 4B;                                   02174
         mlastfaults _ pfaults;                                          02175
         mswdstore(pfaults, 2, 24);                                      02176
         END;                                                            02177
      IF code .A 2B THEN  %time in paging routines%                      02178
         mswdstore(ptime, 3, 18);                                        02179
      END;                                                               02180
   %If truncation error (one of the truncated values lost info)
   record it%                                                            02181
      IF truncerr THEN                                                   02182
         BEGIN                                                           02183
         msrecord(mtruncerr, 0, truncerr);                               02184
         truncerr _ FALSE;                                               02185
         END;                                                            02186
   RETURN;                                                               02187
   END.                                                                  02188
FINISH of AUXCOD                                                          0654
%......query support routines......%                                     01333
   (qport) PROCEDURE(qflg); % Query language initialization. %           01334

      % This is the entry point into the query language. %               01335
      % note that the flag qflg is not used!  Originally used for
      diferentiating between "query" and "NIC Resource Query %          01336
      % If global flag nlparse is FALSE we came from Exec directly and
      exit will be by execute quit; if nlparse is TRUE we came from nls
      and exit will do a jump file return.%
                                                                        01337
      LOCAL STRING com[100], filename[50];                              01338
      ON SIGNAL ELSE                                                    01339
         BEGIN                                                          01340
         crlf();                                                        01341
         typeas($"Error exit from query");                              01342
         IF NOT nlparse THEN %ceq()% RETURN                             01343
         ELSE                                                           01344
            BEGIN                                                       01345
            ququit();     %Jump file return%                            01346
            %GOTO STATE;%                                               01347
            RETURN;                                                     01348
            END;                                                        01349
         END;                                                           01350
      crlf();                                                           01351
      qustart($filename, $com);                                         01352
      RETURN;                                                           01353
      END.
                                                                        01354
   (qustart) PROCEDURE(filename, com);  % Query parser. %               01355

      % This parser accepts a Bring command and the name of one of four
      basic files. Interrogation of a file continues with the Show
```

```
command. Quit causes exit back to EXEC or TNLS. A question mark
will give command language description to user.%
                                                                    01356
LOCAL wkstid, loaded, param1, resptr[10];                           01357
LOCAL STRING apndir[50], tempsr[50];                                01358
REF filename, com, param1;                                          01359
                                                                    01360
% Set up default directory string %                                 01361
   !JSYS gjinf;                                                     01362
   gdname(r2,$tempsr);                                              01363
   *tempsr* _ "<,*tempsr*,">;                                       01364
*apndir* _ "<NETINFO>";                                             01365
*filename* _ "<NETINFO>HELP1";                                      01366
loaded _ FALSE;                                                     01367
crlf();                                                             01368
% load first "help" file %                                          01369
   wkstid _ quloadit( &filename, $apndir);                          01370
LOOP % parsing loop %                                               01371
   BEGIN                                                            01372
   echon();                                                         01373
   ON SIGNAL                                                        01374
      = ofilerr:                                                    01375
         BEGIN                                                      01376
         IF NOT (FIND SF(*filename*) ["<]) THEN % no directory
         name %                                                     01377
            BEGIN                                                   01378
            *apndir* _ *tempsr*;  % append user directory %         01379
            ON SIGNAL ELSE GO TO realerr;                           01380
            wkstid _ quloadit( &filename, $apndir); % Try with
            different directory name %                              01381
            END;                                                    01382
         (realerr):                                                 01383
         IF sysmsg THEN                                             01384
            BEGIN                                                   01385
            typeas(MESSAGE); % there WAS a dir, or failed twice %
                                                                    01386
            crlf();                                                 01387
            typeas($"File not found");                              01388
            sysmsg _ 0;                                             01389
            END;                                                    01390
         *apndir* _ "<NETINFO>"; % restore default directory %
                                                                    01391
         REPEAT LOOP;                                               01392
         END;                                                       01393
      =statesig: REPEAT LOOP;                                       01394
      ELSE;                                                         01395
   crlf();                                                          01396
   typeas($"-");                                                    01397
   CASE inpcuc() OF                                                 01398
      ="S: % Show command %                                         01399
         BEGIN                                                      01400
         echo($"how ");                                             01401
         quinlit(&com);                                             01402
         deblank(&com);                                             01403
         IF loaded THEN % file is present and loaded %              01404
            q1(wkstid, &com)                                        01405
```

```
   ELSE                                                                 01406
      BEGIN                                                             01407
      typeas($"You have not specified a file yet.");                    01408
      END;                                                              01409
   END;                                                                 01410
="B: % Bring (file name) %                                              01411
   BEGIN                                                                01412
   echo($"ring ");                                                      01413
   quinlit(&com);                  % read filename %                    01414
   deblank(&com);                                                       01415
   *filename* _ *com*;                                                  01416
   loaded _ TRUE;                                                       01417
   wkstid _  quloadit( &filename, $apndir);                             01418
   END;                                                                 01419
="D: % data base display %                                              01420
   BEGIN                                                                01421
   echo($"ata Base of user files");                                     01422
   CASE inpcuc() OF                                                     01423
      = CA, = EOL:                                                      01424
         BEGIN                                                          01425
         *filename* _  "<NETINFO>DATABASE";                             01426
         loaded _ TRUE;                                                 01427
         wkstid _  quloadit( &filename, $apndir);                       01428
         END;                                                           01429
      ENDCASE REPEAT LOOP;                                              01430
   END;                                                                 01431
="R: % Resource notebook display %                                      01432
   BEGIN                                                                01433
   echo($"esource Notebook");                                           01434
   CASE inpcuc() OF                                                     01435
      = CA, = EOL:                                                      01436
         BEGIN                                                          01437
         *filename* _  "<NETINFO>RESOURCES";                            01438
         *apndir* _ "<NETINFO>";                                        01439
         loaded _ TRUE;                                                 01440
         wkstid _ quloadit( &filename, $apndir );                       01441
         END;                                                           01442
      ENDCASE REPEAT LOOP;                                              01443
   END;                                                                 01444
="A: % arpanet news display %                                           01445
   BEGIN                                                                01446
   echo($"RPANET NEWS");                                                01447
   CASE inpcuc() OF                                                     01448
      = CA, = EOL:                                                      01449
         BEGIN                                                          01450
         *filename* _  "<HELP>ARPANEWS";                                01451
         *apndir* _ "<NETINFO>";                                        01452
         loaded _ TRUE;                                                 01453
         wkstid _ quloadit( &filename, $apndir );                       01454
         END;                                                           01455
      ENDCASE REPEAT LOOP;                                              01456
   END;                                                                 01457
="I: % Ident file display %                                             01458
   BEGIN                                                                01459
   echo($"dent File");                                                  01460
   CASE inpcuc() OF                                                     01461
```

```
        = CA, = EOL:                                                    01462
            BEGIN                                                       01463
            *filename* _  "<IDENTFILE>IDENTS.MASTER";                   01464
            loaded _ TRUE;                                              01465
            wkstid _ quloadit( &filename, $apndir );                    01466
            END;                                                        01467
        ENDCASE REPEAT LOOP;                                            01468
    END;                                                                01469
='N: % Start over -- NIC command %                                      01470
    BEGIN                                                               01471
    echo($"ic");                                                        01472
    CASE inpcuc() OF                                                    01473
        = CA, = EOL:                                                    01474
            BEGIN                                                       01475
            *filename* _  "<NETINFO>HELP1";                             01476
            loaded _ FALSE;                                             01477
            wkstid _  quloadit( &filename, $apndir);                    01478
            END;                                                        01479
        ENDCASE REPEAT LOOP;                                            01480
    END;                                                                01481
='?: % help, then back to previous file if any %                        01482
    BEGIN                                                               01483
    CASE inpcuc() OF                                                    01484
        = CA, = EOL:                                                    01485
            BEGIN                                                       01486
            crlf();                                                     01487
            wkstid _ quloadit( $"<NETINFO>HELP", $apndir );
                                                                        01488
            IF loaded THEN                                              01489
                BEGIN                                                   01490
                typeas($" Please wait ");                               01491
                wkstid _ quloadit( &filename, $apndir );                01492
                END;                                                    01493
            END;                                                        01494
        ENDCASE REPEAT LOOP;                                            01495
    END;                                                                01496
='H: % help, then back to previous file if any %                        01497
    BEGIN                                                               01498
    echo($"elp");                                                       01499
    CASE inpcuc() OF                                                    01500
        = CA, = EOL:                                                    01501
            BEGIN                                                       01502
            crlf();                                                     01503
            wkstid _ quloadit( $"<NETINFO>HELP", $apndir );
                                                                        01504
            IF loaded THEN                                              01505
                BEGIN                                                   01506
                typeas($" Please wait ");                               01507
                wkstid _ quloadit( &filename, $apndir );                01508
                END;                                                    01509
            END;                                                        01510
        ENDCASE REPEAT LOOP;                                            01511
    END;                                                                01512
%='V:  viewspecs  not needed (show and bring both force
viewspecs)%                                                             01513
    %BEGIN%                                                             01514
```

```
            %echo($"iewspecs: Type ");%                                    01515
            %getvsp(); gets string from keyboard%                          01516
            %treslev(tda.dacsp); take care of relative levels%             01517
            %&param1 _ xviewspecs($resptr,1);                              01518
            cspupdate _ lda();                                             01519
            cspvs _ param1;                                                01520
            cspvs[1] _ param1[1] ;                                         01521
            dpset(dspyes,[cspupdate].dacsp,endfil,endfil);                 01522
            cmdfinish();                                                   01523
            END;%                                                          01524
         ="Q: % quit %                                                     01525
            BEGIN                                                          01526
            echo($"uit ");                                                 01527
            CASE inpcuc() OF                                               01528
               = CA, = EOL:                                                01529
                  BEGIN                                                    01530
                  IF NOT nlparse THEN %ceq()% RETURN                       01531
                  ELSE                                                     01532
                     BEGIN                                                 01533
                     %Jump file return%                                    01534
                     ququit();                                             01535
                     %GOTO oldgps;%                                        01536
                     RETURN;                                               01537
                     END;                                                  01538
                  END;                                                     01539
               ENDCASE REPEAT LOOP;                                        01540
            END;                                                           01541
         ENDCASE                                                           01542
            BEGIN                                                          01543
            crlf();                                                        01544
            typeas($"Not recognized");                                     01545
            END;                                                           01546
      END;                                                                 01547
   END.
                                                                           01548
(quloadit) PROCEDURE( filename, apndir );                                  01549
   LOCAL fileno, wkstid;                                                   01550
   LOCAL STRING qsname[100];                                               01551
   REF filename, apndir;                                                   01552
   *qsname* _ *filename*;                                                  01553
   IF NOT (FIND SF(*qsname*) ['<]) THEN % filename has no directory %
                                                                           01554
      *qsname* _ *apndir*, *filename*;                                     01555
   %xif($qsname, &tda);                                                    01556
   sysmsg _ 0;                                                             01557
   trefint();                                                              01558
   fileno _ tda.dacsp.stfile;%                                             01559
   % --- new code ---%                                                     01560
      fileno _ cloafil($qsname);                                           01561
      curmkr _ orgstid;                                                    01562
      curmkr.stfile _ fileno;                                              01563
      curmkr[1] _ 1;                                                       01564
   *apndir* _ "<NETINFO>";                                                 01565
   crlf();                                                                 01566
   typeas($"--------");                                                    01567
   feedit(&tda,$"Bw");                                                     01568
```

```
   wkstid _ origin;                                                    01569
   wkstid.stfile _ fileno;                                             01570
   IF (wkstid _ getsub(wkstid)).stpsid = origin THEN                   01571
      BEGIN                                                            01572
      typeas($"File is empty");                                        01573
      END                                                              01574
   ELSE                                                                01575
      BEGIN                                                            01576
      feedlt(&tda, $"esb");                                            01577
      printg(&tda,wkstid, wkstid, brnchv,0);                           01578
      END;                                                             01579
   crlf();                                                             01580
   typeas($"--------");                                                01581
   RETURN( wkstid );                                                   01582
   END.
                                                                       01583
(q1) PROCEDURE (orstid, com); %converts "show" to appropriate jump%
                                                                       01584

   % given the string typed by the user, find a statement by that
   name in the current file.%
                                                                       01585
   LOCAL prvstid, wkstid;                                              01586
   LOCAL TEXT POINTER z1, z2, z3, z4;                                  01587
   LOCAL STRING com1[100], erout[100];                                 01588
   REF com;                                                            01589
   %Initialize stids %                                                 01590
      wkstid _ prvstid _ orstid;                                       01591
   feedlt(&tda, $"Bw"); % Don't indent; all lines, all levels %        01592
   *erout* _ NULL;                                                     01593
   IF NOT (FIND SF(*com*) ^z1 [":] ^z2 _z2 ^z3 [ENDCHR] ^z4) THEN

      % User has typed a command like: show xyz CR %
                                                                       01594
      BEGIN                                                            01595
      wkstid _ namingrp(prvstid, prvstid, &com, 1000);                 01596
      IF wkstid # endfil THEN                                          01597
         BEGIN                                                         01598
         guprout( wkstid );                                            01599
         RETURN;                                                       01600
         END                                                           01601
      ELSE                                                             01602
         BEGIN                                                         01603
         wkstid _ namingrp(orstid, endfil, &com, 1000);                01604
         IF wkstid # endfil THEN                                       01605
            BEGIN                                                      01606
            guprout( wkstid );                                         01607
            RETURN;                                                    01608
            END;                                                       01609
         crlf();                                                       01610
         *erout* _ *com*," not found.";                                01611
         typeas($erout);                                               01612
         RETURN;                                                       01613
         END;                                                          01614
      END                                                              01615
   ELSE                                                                01616
```

```
      % User has typed: show xyz:abc CR
      which means: Find a branch named abc anywhere within the branch
      named xyz. %
                                                                    01617
      BEGIN                                                         01618
      *com1* _ z3 z4;                                               01619
      *com* _ z1 z2;                                                01620
      wkstid _ namingrp(orstid, endfil, &com, 1000);                01621
      IF wkstid = endfil THEN                                       01622
         BEGIN                                                      01623
         crlf();                                                    01624
         *erout* _ *com*," not found.";                             01625
         typeas($erout);                                            01626
         RETURN;                                                    01627
         END;                                                       01628
      prvstid _ wkstid;                                             01629
      wkstid _ namingrp(prvstid, prvstid, $com1,1000);              01630
      IF wkstid # endfil THEN                                       01631
         BEGIN                                                      01632
         quprout( wkstid );                                         01633
         RETURN;                                                    01634
         END                                                        01635
      ELSE                                                          01636
         BEGIN                                                      01637
         crlf();                                                    01638
         *erout* _ *com1*, " not found under ", *com*, '.;          01639
         typeas($erout);                                            01640
         RETURN;                                                    01641
         END;                                                       01642
      END;                                                          01643
   END.

                                                                    01644
(quprout) PROCEDURE (stid);                                         01645
   feedit(&tda, $"esb");                                            01646
   qubing(stid);                                                    01647
   printg(&tda,stid, stid, brnchv,0);                               01648
   RETURN;                                                          01649
   END.

                                                                    01650
(qubing) PROCEDURE (stid); % process embedded viewspecs.%           01651

   % When a statement name is followed by a string such as (uv:) this
   procedure recognizes uv as viewspecs and turns them on.%
                                                                    01652
   LOCAL STRING vstrng[10];                                         01653
   LOCAL char;                                                      01654
   *vstrng* _ NULL;                                                 01655
   s2work _ stid;                                                   01656
   s2work[1] _ fchtxt(getsdb(stid));                                01657
   fechc1(forward,$s2work);                                         01658
   IF (char _ READC($s2work)) # '( THEN RETURN;                     01659
   IF (char _ READC($s2work)) # ': THEN RETURN;                     01660
   LOOP                                                             01661
      BEGIN                                                         01662
      char _ READC($s2work);                                        01663
      IF char # ') THEN *vstrng* _ *vstrng*,char                    01664
```

```
      ELSE                                                          01665
         BEGIN                                                      01666
         feedit(&tda, $vstrng);                                     01667
         RETURN;                                                    01668
         END;                                                       01669
      END;                                                          01670
   END.
                                                                    01671
(ququit) PROCEDURE; % Quit and restore nls context. %               01672

   % This procedure is called by the parser when the quit command is
   encountered and entry was from nls.%
                                                                    01673
   LOCAL STRING str[100];                                           01674
   %gadjf($b1, $popls, $str, 1, &tda);%                             01675
   %mvcb1();%                                                       01676
   %dismes(2,$"Ququit Called");%                                    01677
   RETURN;                                                          01678
   END.                                                             01679
                                                                    01680
(quinlit) PROCEDURE(com); %read a string, handle special characters.%
                                                                    01681

   % Uses inpcuc iteratively until it finds CA or EOL (in which case
   it returns TRUE). Handles CD and BC normally. Question mark forces
   the string 'H into com. This can be used as third-level help but
   no current file takes advantage of it. Feature could be taken out
   without effect on other procedures.%
                                                                    01682
   LOCAL char;                                                      01683
   REF com;                                                         01684
   LOOP                                                             01685
      BEGIN                                                         01686
      *com* _ NULL;                                                 01687
      LOOP                                                          01688
         BEGIN                                                      01689
         char _ inpcuc();                                           01690
         CASE char OF                                               01691
            =BC: IF com.L > empty THEN                              01692
               BEGIN                                                01693
               todco(*com*[com.L]);                                 01694
               bkc(&com);                                           01695
               END;                                                 01696
            =CD: SIGNAL(statesig, 0);                               01697
            =CA: RETURN(TRUE);                                      01698
            =EOL: RETURN(TRUE);                                     01699
            ='?:                                                    01700
               BEGIN                                                01701
               *com* _'H ;                                          01702
               RETURN(TRUE);                                        01703
               END;                                                 01704
            ENDCASE *com* _ *com*, char;                            01705
         END;                                                       01706
      END;                                                          01707
   END.
                                                                    01708
```

```
(deblank) PROCEDURE (string);  %deblank the string%                    01709

   % Eliminate leading blanks for show and bring command. %
                                                                       01710
   LOCAL TEXT POINTER z1, z2;                                          01711
   REF string;                                                         01712
   IF FIND SF(*string*) ^z1 $NP ^z2 THEN                               01713
      ST z1 z2 _ NULL;                                                 01714
   RETURN;                                                             01715
   END.                                                                01716
```

BCONST

```
< NLS, BCONST.NLS.50, >, 4-Dec-78 19:04 JCP ;;;;

FILE bconst    % L10  <REL-NLS>bconst %% (L10,) (rel-nls,bconst.rel,) %
                                                                      02
%   MAP OF NLS SYSTEM--                                               03
   Pages 0  -- 377:                                                   04
      Low segment of code is loaded up from page 0; the symbol table is
      loaded down from page 400.  If they overlap, we are in trouble.
      (Also, the loader someties blows up in mysterious ways in this
      case!)  When a user program is loaded, a mark is placed at the end
      of the symbol table and the user program symbol's are placed down
      from that.  Thus it pays to have some room there.  The code in
      this segment is generally core routines.                        05
   Pages 400 -- 553:                                                  06
      High segment is used for the CML parser and interpreter code as
      well as parse support code (X- routines, etc.)  When processors
      (compilers, output processors) are run, this code is swapped out
      and the processors mapped into this area.  Upon completion, this
      code is brought back.  (See (nls, seqfil, processor).)  Page 552
      is used for the user-option system ( definition of this page is in
      (nls,updata,) ).  Page 553 is used for mapping pages out in Output
      File.  These two pages must not be overwritten by code!         07
   Pages 554 -- 653:                                                  08
      These pages are used for the User Program Buffer area (lower part)
      and for the File buffer area (upper part).  The boundary may be
      moved by the user using the Set Buffer command in the Program
      subsystem.  The default size of the Program Buffer is 4 pages.  09
   Pages 654 -- 657:                                                 010
      Currently unused.  May be used to extend the size of the loader
      hash table area.                                               011
   Pages 660 -- 677:                                                 012
      Loader hash table area.  Filled from 677777B down.             013
   Page 700:                                                         014
      Used by the loader.                                            015
   Pages 701 -- 704                                                  016
      Sequence stack area. Two stacks of 2 pages each                017
   Pages 705 -- 757                                                  018
      Storage allocator space. 53 Octal (43 decimal) pages used for lsrt
      tables, string storage, sequence generator stacks, and sequence
      generator statement vector work areas.                         019
   Pages 760 -- 764:                                                 020
      Loader.                                                        021
   Pages 765 -- 767:                                                 022
      Currently unused.                                              023
   Pages 770 -- 777:                                                 024
      DDT.                                                           025
%
                                                                      026
SET EXTERNAL  % locations in high core %                              027
   dspblk= 705000B,  % beginning of storage allocator block. %        028
      % used to store lsrt's, strings from getstring, etc. %         029
   dspbke= 757777B;  % end of storage allocator block. %             030
SET EXTERNAL  % locations in high core %                              031
   btree= 400000B,                                                   032
   efree= 554000B,                                                   033
      % these pages are currently used for:                          034
```

```
        compilers                                                       035
        Output Processor                                                036
        Update Checkpoint, etc.                                         037
        load 940 files                                                  038
        Journal Delivery                                                039
        Sort/Merge %                                                    040
    ckpag= 553000B, % page used in updtfl as work area %                041
    sqstks= 701000B;  % 2 Sequence stacks of 2000B each to 704777B %    042
                                                                        043
SET EXTERNAL sqstkn = 2; %number of declared sequence stacks (others may
be allocated from dspblk)%                                              044
DECLARE EXTERNAL                                                        045
    dpybks= 53000B;  % Size of storage allocation block.  53 Octal (43
    Decimal) pages. %                                                   046
% error codes -- used in SIGNALling %                                   047
    % errors generated by link parsing %                               048
        DECLARE EXTERNAL                                                049
            % lnk1err= 9649871,                                        050
                Illegal Link Syntax:
              ENDCHR Before Closing " %                                 051
            lnk2err= 9649872,                                           052
              % Illegal Link Syntax:
              ENDCHR Before CH After ' %                                053
            % lnk3err= 9649873,                                        054
                Illegal Link Syntax:
              Illegal Statement Name or Number After ! or * %           055
            % lnk4err= 9649874,                                        056
                Illegal Link Syntax:
              Illegal DAE Element %                                     057
            lnk5err= 9649875,                                           058
              % Illegal Link:
              Left Delimeter Not Found %                                059
            lnk6err= 9649876,                                           060
              % Illegal Link Syntax or Semantic:
              Missing Right Delimeter or Bad Viewspecs %                061
            lnk7err= 9649877,                                           062
              % Illegal Link Syntax:
              Missing Right Delimeter %                                 063
            % lnk8err= 9649878,                                        064
                Illegal Link Syntax:
              Illegal Marker After # %                                  065
            lnk9err= 9649879                                            066
              % Illegal Link Syntax:
              ENDCHR Before ; In Filter %                               067
            ;                                                           068
    DECLARE EXTERNAL                                                    069
        bfile = 4,              %I/O errors to err%                     070
        clisterr= 7364444B,     % correspondence list overflow %       071
        sqerr=   7369370B,      % sequence generator error code %      072
        upgerr=  4356712B,      % user program error code %             073
        sorterr= 4624894B,      % sort-merge error %                    074
        spacerr= 7360725B,      %display support failed to get space%  075
        % freerr= 7360726B,      display support failed to free space% 076
        ofilerr= 74563342B,     % can't open a file %                   077
        prcerr=  221124768;     % can't get a jfn for a processor %     078
    DECLARE EXTERNAL                                                    079
```

```
    werrsig= -1;                                                    080
  SET EXTERNAL ftpsig= ofilerr;                                     081
                                                                    082
SET EXTERNAL %character codes and group names%                      083
  %group names, for use with define state pop (ds)%                 084
    spec=  1,   %special group%                                     085
    edit=  2,   %edit group%                                        086
    jump=  3,   %jump group%                                        087
    vect=  4;   %vector package group%                              088
% File Things %                                                     089
  DECLARE EXTERNAL %for global symbolic constants%                  090
    rfpmax=1008, %number of core pages for file blocks%             091
    hedbas=0,    %first file page for header blocks%                092
    fust0=filhed,  %address of short link default string%           093
    fust1=filhd1,  %address of long link default string%            094
    rngbas=6,    %first file page for ring blocks%                  095
    rngm=95,     %max number of ring blocks%                        096
    dtbbas=101,  %first file page for data blocks%                  097
    dtbm=370,    %max number of data blocks%                        098
    ringl=5,     %length of ring element%                           099
    fbhdl=2,     %length of file block header%                      0100
    swrkl=5,     %length of work area for fechc1%                   0101
    hdtyp=0,     %block type for header blocks%                     0102
    sdbtyp=1,    %block type for sdb blocks%                        0103
    rngtyp=2,    %block type for ring blocks%                       0104
    jnktyp=3,    %block type for misc blocks%                       0105
    niltyp=4,    %block type for blocks to be initialized%          0106
    blksiz=512;  %random file block size%                           0107
  DECLARE EXTERNAL % System data block property types %             0108
    % Users wishing to have special properties in their programs need
    not declare the properties here unless they are to be incorporated
    into the NLS system or unless they wish to avoid conflict with
    other special applications.  The properties reserved for such
    experimental user applications are 40000B-77777B.  Declared
    properties may be used for other purposes only if the programmer
    is sure there will be no conflict (such is currently the case with
    QUERY3 making use of some of the graphics types.)  This practice
    is NOT recommended!  If a property does not fall in the USER
    range, it must be assigned an order through inclusion in the table
    PROPTAB.  See (nls, filmnp, locprop) and (nls, filmnp, getptab). %
                                                                    0109
      txttyp= 0, % text data block %                                0110
      gtftyp= 1, % graphics text format data block %                0111
      lwtyp= 2, % line work data block %                            0112
      chtyp= 3, % cell header data block %                          0113
      dhtyp= 4; % diagram header data block %                       0114
  DECLARE EXTERNAL % data block property location table %           0115
    proptab= (5 % length of property table-- number of properties %,
                                                                    0116
      1, % gtftyp %                                                 0117
      0, % txttyp %                                                 0118
      2, % lwtyp %                                                  0119
      3, % chtyp %                                                  0120
      4); % dhtyp %                                                 0121
    % defines the order in which system properties will occur in the
    statement.  Used to prevent the necessity of searches of all
```

```
   properties.  Need not be used for special user properties.  See
   comment above with the definition of the property types. %     0122
DECLARE EXTERNAL                                                  0123
   crpgad=(                                                       0124
      %core locations for file block pages%                       0125
      % indexed by numbers in [rfpmin,rfpmax] %                   0126
      0 %filler%,                                                 0127
      %100B pages in high core%                                   0128
         554000B, 555000B, 556000B, 557000B,                      0129
         560000B, 561000B, 562000B, 563000B, 564000B, 565000B,
         566000B, 567000B,                                        0130
         570000B, 571000B, 572000B, 573000B, 574000B, 575000B,
         576000B, 577000B,                                        0131
         600000B, 601000B, 602000B, 603000B, 604000B, 605000B,
         606000B, 607000B,                                        0132
         610000B, 611000B, 612000B, 613000B, 614000B, 615000B,
         616000B, 617000B,                                        0133
         620000B, 621000B, 622000B, 623000B, 624000B, 625000B,
         626000B, 627000B,                                        0134
         630000B, 631000B, 632000B, 633000B, 634000B, 635000B,
         636000B, 637000B,                                        0135
         640000B, 641000B, 642000B, 643000B, 644000B, 645000B,
         646000B, 647000B,                                        0136
         650000B, 651000B, 652000B, 653000B);                     0137
DECLARE EXTERNAL STRING  %directory names and file extensions%    0138
   prtdir= "ARCPRINTER", %may be changed by (nls,fintnls,saveit)%
                                                                  0139
   comdir= "COM",                                                 0140
   relext= "REL",                                                 0141
   txtext= "TXT";                                                 0142
DECLARE EXTERNAL                                                  0143
   nvtk= 2,          %number of versions of a file to keep%       0144
   n40fil= 0,                                                     0145
   tenfil= 1,        %normal sequential file (1 CR to end stmts)%
                                                                  0146
   macfil= 2,        %MACRO file (assembled)%                     0147
   assfil= 3,        %assembler file with structure%              0148
   heurfil= 88,      %sequential file with 2 CRs to end stmts%    0149
   justfil= 89,      %seq file w/ 2 CRs to end stmts, and justified%
                                                                  0150
   opsqfl=1,         %output type = sequential file%              0151
   opqpfl=2,         %output type = quickprint file%              0152
   opmcfl=3,         %output type = assembler file%               0153
   opmlfl=4,         %output type = mail file%                    0154
   chkpcf= TRUE,     %PC being opened; access only by this fork% 0155
   origff= FALSE,    %original file (not PC); access by others OK%
                                                                  0156
   oldvrsn= -4,      %use the highest existing version of the file%
                                                                  0157
   dfltvrs= FALSE;   %use the next higher version of the file%    0158
SET EXTERNAL                                                      0159
   pspublic= 1,                                                   0160
   psprivate= 2;                                                  0161
% Processors %                                                    0162
DECLARE EXTERNAL                                                  0163
% types of output processors %                                   0164
```

```
    upgtyp= 1,   % compile a user program into the buffer %          0165
    cmptyp= 2,   % compile a program into a REL file %               0166
    odtyp = 3,   % output device something %                         0167
  prcadr= 400010B;  % a call [[prcadr]] gets to processor %         0168
% Sequence Generator %                                              0169
  DECLARE EXTERNAL                                                  0170
    % tells user sequence generators what they are being called as %
                                                                    0171
    sqopn= 1,                                                       0172
    sqcls= 2,                                                       0173
    sqgnxt= 3;                                                      0174
% VALUES FOR MODES %                                               0175
  % all multiply used values should be defined here %               0176
  SET EXTERNAL                                                      0177
    vexpert= 1,                  % expert %                         0178
    vfullprompts= 3,  % full prompting %                            0179
    vtersemode= 1,    % terse mode %                                0180
    vmultchar= 2,     % multiple character hearald %               0181
    vverbsmode= 2;    % verbose feedback %                         0182
% RECORD SIZE DECLARATIONS %                                       0183
  DECLARE EXTERNAL                                                  0184
    pvslen= 3,        % length of pvsrecord (offset to viewspec
    collection astring %                                            0185
    plvlen= 1,        % length of plvrecord (offset to leveladj
    collection astring %                                            0186
    pconlen= 1,       % length of pconrecord %                     0187
    pkeylen= 1;       % length of pkeyrecord %                     0188
% LEVEL ADJUST VALUE CODES %                                       0189
  DECLARE EXTERNAL                                                  0190
    levsuc= 0,        % value for successor %                      0191
    levup= 1, % value for up one level %                            0192
    levdown= -1;      % value for down one level %                 0193
% description of the data structure for parsed links %            0194
  DECLARE EXTERNAL                                                  0195
    lfn= 0,   % file number for default directory %                  0196
    ls= 1,    % text pointer to get start of link %                 0197
    le= 3,    % text pointer to get end of link %                   0198
    cs= 5,    % text pointer to get start of comment %              0199
    ce= 7,    % text pointer to get end of comment %                0200
    hs= 9,    % text pointer to get start of hostname %             0201
    he= 11,   % text pointer to get end of hostname %               0202
    us= 13,   % text pointer to get start of username %             0203
    ue= 15,   % text pointer to get end of username %               0204
    fs= 17,   % text pointer to get start of filename %             0205
    fe= 19,   % text pointer to get end of filename %               0206
    ds= 21,   % text pointer to get start of dae %                  0207
    de= 23,   % text pointer to get end of dae %                    0208
    vb= 25,   % text pointer to get start of viewspecs %           0209
    ve= 27,   % text pointer to get end of viewspecs %             0210
    lnkdsl= 29;       % length of this data structure %            0211
% user-options data page layout definitions %                      0212
  % the declarations for the user-profile page are contained in file
  updata %                                                          0213
% user-options defaults %                                          0214
  DECLARE EXTERNAL                                                  0215
  % control characters %                                           0216
```

```
% current context %                                                    0217
   defcurcontext=7,  %no. of characters to either side of CM for /%
                                                                      0218
% display %                                                           0219
   defdcolmax= 72,   % max no. of printing chars to display%          0220
   defdwrapcol= 0,   % default value for no wrapping to occur %       0221
% feedback %                                                          0222
   defbmode= vverbsmode, % feedback mode %                            0223
   defdind= 0,       % typewriter feedback indentation%              0224
   defdbk= 50,       % number of character of feedback to echo%      0225
% hearald %                                                           0226
   defhmode= vmultchar,      % command hearalding mode %             0227
   defhsize= 4,      % # chars in multchar hearald %                  0228
% jump rings %                                                        0229
   defsrrsize=10,    %no. entries in file return ring%               0230
   deffrrsize=10,    %no. entries in statement return ring%          0231
% print options %                                                     0232
   defpgsize= 66,    % number of lines per page%                     0233
   deflinmax= 60,    % max no. of lines to print per page%           0234
   defcolmax= 72,    % max no. of columns to print%                  0235
   defindcnt= 3,     % indenting per level%                          0236
   defoffset= 0,     % number of columns to offset left margin%      0237
   deftb1= 7767757737678, % default tab settings( 8,16,24,...) %     0238
   deftb2= 7577376775778,                                            0239
   deftb3= 3767757737678,                                            0240
% prompt %                                                           0241
   defprompt= vfullprompts,  % default full prompts%                  0242
% quickprint %                                                       0243
   defqcolmax= 72,   % max no. of columns to print%                   0244
% recognition %                                                      0245
   defrecmode= vexpert,                                              0246
   defre2mode= vexpert,                                              0247
% space for tab values.  See printoptions for tabstops %            0248
   defspftab= 0,                                                     0249
   defrtjtab= 0,                                                     0250
   defrjtchr=11B,                                                    0251
% null default string for initial process commands %                 0252
   defstu= 310B6,    % 200 word null string %                        0253
% null default string for external names link file %                 0254
   defenlf= 310B6,   % 200 word null string %                        0255
% null default string for user profile subsystems and programs%      0256
   defsubstr= 36B6;                                                  0257
% default subsystems and programs to include at initialization%      0258
   DECLARE EXTERNAL STRING                                           0259
      dsys1= "SUPERVISOR",                                           0260
      dsys2= "BASE",                                                 0261
      dsys3= "USEROPTIONS",                                          0262
      dsys4= "SENDMAIL",                                             0263
      dsys5= "PROGRAMS",                                             0264
      dsys6[30],                                                     0265
      dsys7[30],                                                     0266
      dsys8[30],                                                     0267
      dsys9[4],                                                      0268
      dsys10[4],                                                     0269
      dsys11[4],                                                     0270
      dsys12[4],                                                     0271
```

```
        dsys13[4],                                                      0272
        dsys14[4],                                                      0273
        dsys15[4];                                                      0274
% user programs buffer size calculations %                             0275
   SET EXTERNAL                                                         0276
      upgbdf= 6;  % default size for user programs buffer%              0277
   DECLARE EXTERNAL                                                     0278
      maxbsize= 44,  %maximum user programs buffer size%                0279
      minfpages= 20;  %minimum no of file pages allowed%                0280
% Collecter sorter %                                                   0281
   SET EXTERNAL                                                         0282
      v=      400000B,    %address of vector used for sorting%          0283
      vcvu=   400100B,                                                  0284
      vcvul= 400070B,                                                   0285
      vcv=    420000B;    %primary sort key value%                      0286
   DECLARE EXTERNAL                                                     0287
      vmax=    20000B,     %max length of sort vector%                  0288
      vcvend= 477750B;    %end of sort key value area%                  0289
                                                                        0290
% Level adjust Parameters %                                           0291
   DECLARE EXTERNAL STRING                                              0292
      downstr=  "d",                                                    0293
      sucstr[0];                                                        0294
   DECLARE EXTERNAL                                                     0295
      sucdir= 0,          %fake first word of a string (addr passed to
      levset to get level art for a successor)%                         0296
      down= -1;           %  not used??  %                              0297
% Structure type definitions %                                         0298
   DECLARE EXTERNAL                                                     0299
      stmtv= 1,           %for printing and journal message source%     0300
      brnchv=2,           %for printing%                                0301
      groupv=3,           %for printing and journal message source%     0302
      plexv=4,            %-----not used??-----%                        0303
      litv= 5,            %-----not used??-----%                        0304
      filev= 6,           %for journal message source%                  0305
      hcopyv= 7,          %for journal message source%                  0306
      charv= 8,           %for substitute%                              0307
      wordv= 9,           %for substitute%                              0308
      numbrv= 10,         %for substitute%                              0309
      visv= 11,           %for substitute%                              0310
      invisv= 12,         %for substitute%                              0311
      linkv= 13,          %for substitute%                              0312
      textv= 14,          %for substitute%                              0313
      tdfltv= 15,         %for substitute%                              0314
      forwdv= 16;         %for journal forwarding of mail%              0315
% File type definitions %                                              0316
   DECLARE EXTERNAL                                                     0317
      bintyp= 2,          %binary file; 36-bit bytes%                   0318
      chrtyp= 3,          %character file; 7-bit bytes%                 0319
      random= 4,          %random file; no byte I/O, PMAPs only%        0320
      comtyp= 5,          %com file; 7-bit bytes%                       0321
      lpttype= 6,         %line printer file; 7-bit bytes%              0322
      netype= 7;          %network file in buffered send mode; 8-bit
      bytes%                                                            0323
% Lookup Types %                                                       0324
   DECLARE EXTERNAL                                                     0325
```

```
      nametyp= 1,       %fast name lookup from start of file%       0326
      wordtyp= 2,       %word search%                               0327
      contnt= 3,        %content search%                            0328
      sid= 4,           %sid search%                                0329
      cont1s=6, %single statemen content search%                    0330
      nxtname= 7,       %fast name loopup from specified STID%      0331
      seqname= 8,       %sequential name lookup%                    0332
      braname= 9,       %name lookup within specified branch%       0333
      extname= 10,      %name lookup - on failure take link in sysgd%
                                                                    0334
      word1s= 11;       %single statement word lookup%              0335
% Control Character echoing control %                               0336
   DECLARE EXTERNAL                                                 0337
      ttycoc= 10735B3,                                              0338
      dpycoc= 104012537B3;                                          0339
% Subsystem names %                                                 0340
   DECLARE EXTERNAL STRING                                          0341
      opname= "<NETSYS>NOUTPRC.SAV
      ", %this is changed to <SUBSYS> by (fintnls,saveit) for
      "non-standard" hosts%                                         0342
      xopname= "<NETSYS>XOUTPRC.SAV
      ",                                                            0343
      idnfname= "<IDENTS>IDENTS.NLS", %changed by (fintnls,saveit) for
      some hosts%                                                   0344
      jnlnam= "JOURNL",                                             0345
      l10nam= "L10",                                                0346
      xl10nam= "XL10",                                              0347
      %trenam= "TREEMETA",%                                         0348
      subdir= "NETSYS",                                             0349
         %directory for compilers, MSGNLS%                          0350
      subdr2= "SUBSYS";                                             0351
% Identification System %                                          0352
   DECLARE EXTERNAL                                                 0353
      indtyp= 1,        %individual ident record%                   0354
      grptyp= 2,        %group ident record%                        0355
      orgtyp= 3,        %organization ident record%                 0356
      afgtyp= 4;        %-----not used??-----%                      0357
% Device things %                                                  0358
   DECLARE EXTERNAL                                                 0359
   % devices for Output Processor %                                 0360
      opsqdv= 1,                % sequential file %                 0361
      optydv= 2,                % typewriter %                      0362
      opcmdv= 3,                % COM %                             0363
      opprdv= 4,                % Printer %                         0364
      opdrdv= 5,                % DURA? %                           0365
      opxpdv= 6,        % COM test %                                0366
      opxtdv= 7,                                                    0367
      oprmdv= 10;               % remote Printer/Terminal %         0368
%imlac protocal%                                                   0369
   DECLARE EXTERNAL %control codes for remote displays (e.g. imlacs)%
                                                                    0370
      remada= 1, %alocate display area control character%           0371
      remdda= 2, %dealocate display area control character%         0372
      remstr= 4, %string manipulation control character%            0373
      remscsr= 5, %set cursor string control character%             0374
      remssda= 10B, %suppress string in display area control character%
```

Host ident record?

```
                                                                    0375
    remscm= 15B, %switch to short character mode%                   0376
    nwada= 16B, %new ada%                                           0377
    apsda= 17B, %append string to seq da%                           0378
    nwstrda= 20B, %new strda%                                       0379
    ccnda= 21B, %start continuous transmision of cursor coords%     0380
    ccfda= 22B, %stop continuous transmision of cursor coords%      0381
    reminit= 33B; %initialize all display areas control character%
                                                                    0382
% Display stuff %                                                  0383
  DECLARE EXTERNAL                                                  0384
    fontmax= 512,                                                   0385
    fnormal= 0;                                                     0386
  % protocol codes  and other declarations %                       0387
    DECLARE EXTERNAL  % Line processor protocol codes %             0388
      % input semantic codes in big characters %                   0389
        fcor2= 42B, % 2 coordinates not with a character %          0390
        fcor4= 44B, % 4 coordinates not with a character %          0391
        cnch2= 41B, % control char %                                0392
        acor2= 43B, % 2 coordinates with M.B. or C.C. %             0393
        acor4= 45B, % 4 coordinates with a character %              0394
        irep= 46B,  % interrogate response %                        0395
      cooresc= 36B, %big coordinate escape, used in both directions%
                                                                    0396
        %the coordinate value follows in two characters, six bits
        each%                                                       0397
      lpesc= 33B,             % escape code to imlac/LP %          0398
      lpresc= 34B,            % escape code from imlac/LP %        0399
      lpposition= 40B,        % position code %                    0400
      lptty= 41B,             % specify tty window code %          0401
      lptrack= 42B,  % resume tracking code %                      0402
      lpcline= 43B,  % clear line code %                          0403
      lpdline= 44B,  % delete line code %                         0404
      lpinline= 45B, % insert line code %                         0405
      lpbug= 46B,             % bug selection code %               0406
      lppbug= 47B,            % pop bug selection code %           0407
      lpcscreen= 50B,         % clear screen code %                0408
      lpreset= 51B,  % reset code %                               0409
      lpintter= 55B, % interrogate code %                         0410
      lpstandout= 56B,        % stand out code %                   0411
      lpendstandout= 57B,     % end stand out code %               0412
      lpnocoor= 60B, % exit coordinate mode %                     0413
      lpcoord=61B,            % enter coordinate mode %            0414
      lptptopen=53B, % LP terminal printer open %                 0415
      lptptclose=54B,         % LP terminal printer close %        0416
      lptptsr=52B,   % LP terminal printer string %               0417
      lptnptsr=63B,  % new LP terminal printer string%            0418
      lptnopn=64B,  % new LP terminal printer open %              0419
      lpscroll=65B;  % new LP terminal scroll window %            0420
                                                                    0421
  %strings used to display non-printing characters - text-pointers in
  isrt point here.  Note that CR and EOL are used only by display name,
  and are interpreted properly by literal dislay and stfrmt%        0422
                                                                    0423
      DECLARE EXTERNAL STRING                                       0424
        castr= "<CA>", cdotstr= "<C.>", lfdstr= "<LF>", cdstr=
```

```
          "<CD>", bcstr= "<BC>", escstr= "<ESC>", astr= "<^A>", bstr=
          "<^B>", cstr= "<^C>", estr= "<^E>", fstr= "<^F>", gstr=
          "<^G>", hstr= "<^H>", kstr= "<^K>", lstr= "<^L>", mstr=
          "<CR>", nstr= "<^N>", ostr= "<^O>", pstr= "<^P>", qstr=
          "<^Q>", rstr= "<^R>", sstr= "<^S>", tstr= "<^T>", ustr=
          "<^U>", vstr= "<^V>", wstr= "<^W>", xstr= "<^X>", ystr=
          "<^Y>", zstr= "<^Z>", uslstr= "<^/>", ubrstr= "<^]>", uustr=
          "<^^>", crstr= "<CR>", enlstr= "<EOL>", tbstr= "<TAB>",
          nulstr= "<NUL>";                                          0425
   %correspondence tables for non-printing character display strings%
                                                                    0426
   %codes encountered in user files%                                0427
                                                                    0428
      DECLARE EXTERNAL                                              0429
         npcodes= (CA, C., LF, CD, BC, 33B, 1B, 2B, 3B, 5B, 6B, 7B, 10B,
         13B, 14B, 15B, 16B, 17B, 20B, 21B, 22B, 23B, 24B, 25B, 26B,
         27B, 30B, 31B, 32B, 34B, 35B, 36B, CR, EOL, TAB, 0B);      0430
                                                                    0431
      %addresses of strings used to disply those codes%             0432
      DECLARE EXTERNAL                                              0433
         npstrings= (castr, cdotstr, lfdstr, cdstr, bcstr, escstr, astr,
         bstr, cstr, estr, fstr, gstr,  hstr, kstr, lstr, mstr, nstr,
         ostr, pstr, qstr, rstr, sstr, tstr, ustr, vstr, wstr, xstr,
         ystr, zstr, uslstr, ubrstr, uustr, crstr, enlstr, tbstr,
         nulstr);                                                   0434
                                                                    0435
      DECLARE EXTERNAL                                              0436
         npsize= 36; %size of non-printing char tables%             0437
% Storage management %                                             0438
   DECLARE EXTERNAL                                                 0439
      blksuc= 1,     %address of successor block in freelist%       0440
      blkprd= 2,     %address of predecessor block n freelist%      0441
      bhl= 1;        %length of an in-use block  header%            0442
% Copy and Show Directory storage management %                     0443
   DECLARE EXTERNAL                                                 0444
      oldpgsz= 0;                                                   0445
% Journal System%                                                  0446
   DECLARE EXTERNAL STRING                                          0447
      jnlpsw[15]; %To be filled in when a system is made%           0448
   DECLARE EXTERNAL %on-line startups%                              0449
      jnitime= (1200170540B, 7021334B, 0, 011500B, 012200B, 012500B, 0,
      0, 0, 0, 0, 0, 0, 0);                                         0450
       %format described in <RECFIL>setojtime , RECORDS at
      <UTILTY>ojtime. 4000010100B, 0, causes no delivery or slinker%
                                                                    0451
% Network/Host dependent %                                         0452
   DECLARE EXTERNAL                                                 0453
      archost=53B,                                                  0454
      utilhost=153B,                                                0455
      bbnbhost=61B,                                                 0456
      sriai20host=102B,                                             0457
      sriaikahost=63B,                                              0458
      isichost=226B,                                                0459
      isidhost=326B,                                                0460
      isir1host=43B,                                                0461
      isiehost=164B,                                                0462
```

```
      nsahost=101B,                                                    0463
      office1host=53B,                                                 0464
      rfchost=153B; %host number where RFC system lives%               0465
   DECLARE EXTERNAL STRING                                             0466
      arcistr= "SRI-ARC", %if ident file says this, then%              0467
      arcstr= "OFFICE-1", %deliver here%                               0468
      nsastr="NSF",  %name for nsa host%                               0469
      bbnbacc= "100842",                                               0470
      utilistr= "OFFICE-1", %if ident file says this, then%            0471
      utilstr= "OFFICE-2"; %deliver here%                              0472
   DECLARE EXTERNAL                                                    0473
      inremsiteflag= TRUE, %may be changed by (nls,fintnls,saveit)%    0474
      outremsiteflag= TRUE; %may be changed by (nls,fintnls,saveit)%
                                                                       0475
% TENEX/TOPS20 dependent %                                            0476
   DECLARE EXTERNAL                                                    0477
      tvrdchar='; ;                                                    0478
         %changed to '. for tops20 at (nls,fintnls,saveit)%            0479
   DECLARE EXTERNAL                                                    0480
      cpcldelim='(, %changed to "$" for tops20 (and ISID)%             0481
      cpcrdelim=');  %changed like cpcldelim%                          0482
   DECLARE EXTERNAL                                                    0483
      t2fsbegin=3;                                                     0484
   DECLARE EXTERNAL STRING %file name strings which get changed for
   tops20 by (nls,fintnls,saveit)%                                     0485
      dirsfname="*.*;*",                                               0486
      flgfname="<NETSYS>NLSFLAGS.FLAGS;1",                             0487
      grpfname="<NETSYS>GROUP.INDEX;1";                                0488
   DECLARE EXTERNAL                                                    0489
      t2fsend=0; %end of filename strings%                             0490
% FTP %                                                               0491
   SET EXTERNAL                                                        0492
      emrtnfalse= 1,                                                   0493
      emsignal= 2,                                                     0494
      emreset= 3;                                                      0495
% Use measurement%                                                    0496
   DECLARE EXTERNAL  %for use measurement%                             0497
      mstxte=0, msstre=1, mschrt=2, mswsiw=3, msstart=1, msfinish=0;
                                                                       0498
   DECLARE EXTERNAL  %for use measurement%                             0499
      msbk11=3, msbk21=4, mshdr1=2, msbavp=30,                         0500
      msrec1=35, msbuf1=512, msdsys=0, mstsys=1;                       0501
   SET EXTERNAL %for offsets into measurement buffer%                  0502
      msproc=0, msreal=1, mscount=2, mspmtot=3, mscurt=0, mssys=1,     0503
      mssmpsz=0, msavgno=1, msavrp=2, msavpf=3, msavws=4;              0504
% Group allocation%                                                   0505
   %user block flag definitions%                                       0506
      DECLARE EXTERNAL                                                 0507
         usroff=400000B,        %this user is off quota%               0508
         usrnot=200000B,        %user has been notified of logout%     0509
         usrexp=100000B,        %job is express%                       0510
         usrlgd=40000B,         %this user is logged in%               0511
         usrina=20000B;                                                0512
      %core page displacement defs%                                    0513
      DECLARE EXTERNAL                                                 0514
         grpadr=3B,                                                    0515
```

```
        expadr=103B,                                                    0516
        ptradr=203B,                                                    0517
        corlgd=304B,                                                    0518
        corlge=305B,                                                    0519
        corexr=307B,                                                    0520
        corblk=316B;                                                    0521
DECLARE EXTERNAL %create display opcodes%
    dspyes= 0,%reformat only da's involved in change%
    dsprfmt= 1,%reformat explicit stids%
    dspstrc= 2,%restructure only%
    dsprfst= 3,%dsprfmt + dspstrc%
    dspjpf= 4,%jump command - no editing%
    dspno= 5;%do not reformat any da's%
                                                                        0522
DECLARE EXTERNAL                                                        0523
   pred=1,       %file structure direction:  predecessor%               0524
   sub=2,        %file structure direction:  down (sub)%                0525
   suc=3,        %file structure direction:  successor%                 0526
   up=4,         %file structure direction:  up%                        0527
   svmxlev=64, % if this is changed, change the dimensions of the arrays
   sqsvws and fvstk %                                                   0528
   alphbase=27, %base for the alpha part of a stmt number%              0529
   alphoff="@,  %offset for the alpha part of a stmt number%            0530
   numbase=10,  %base for the numeric part of a stmt number%            0531
   numoff="0;   %offset for the numeric part of a stmt number%          0532
                                                                        0533
DECLARE EXTERNAL                                                        0534
   upcase=40B,  %set case of characters to all upper%                   0535
   lowcase=1,   %set case of characters to all lower%                   0536
   iupcase=2;   %set case of characters to initial upper%               0537
DECLARE EXTERNAL STRING                                                 0538
   rnmdel= " ";                                                         0539
DECLARE EXTERNAL                                                        0540
   nmalt=5;                                                             0541
      % for use with syntax generating commands %                       0542
        % if a rule has more than nmalt paths, then the rule will not
        be expanded and the name of the rule will appear %              0543
FINISH                                                                  0544
```

B DATA

```
< NLS, BDATA.NLS.19, >, 17-Apr-78 18:27 JDH ;;;;
FILE bdata   %  L10   <REL-NLS>BDATA   %% (L10,) (rel-nls,BDATA.rel,) %

REGISTER %here so DDT will use these on printout%                          01080
   r1=1, r2=2, r3=3, r4=4, r5=5, p=7, wa=8, s=9, m=10, rp=11,                 03
   a1=12, a2=13, a3=14, a4=15;                                               04
% stack sizes %                                                             05
   SET EXTERNAL                                                            06
      %call and multiple result stack lengths%                             07
         groupx=  4,     %maximum group number%                           0963
      %Number of pages to be mapped out for processors%                   0964
         freesz= 154B,                                                     010
      %substitute%                                                         011
         subnum= 30,                                                       012
         subnm1= 120, %=lcard*subnum -- see (UTILTY, card)%                013
      % for stack of user program addresses %                             014
         upgsksz= 20;                                                      015
% *** GLOBAL DATA FOR PAGE 0 *** %                                         016
% Special stuff declared in page 0, loaded at 140B %                      0852
   %...KEEPTHESE THINGS TOGETHER -- DONT ADD NEW DATA HERE...%            0859
   DECLARE EXTERNAL %...important stuff...%                               01082
   %...file block IO...%                                                  0861
      rfpmin= 7,          % Index of lowest file page this constant must 0865
      be changed when the default buffer size is changed %                0866
      nxpg,              %index into corpst for next page%                0867
      corpst[100B],      %core page status table%                         0868
      filehead[26],      %array of filhdr(fileno) values%                 0869
      filepart[26],      %array of -- TRUE if partial copy for file%      0870
   %...often used work areas...%                                          0871
      s2work[7], % second string reading work area %                      0874
   %...seqgen flags...%                                                   0877
      inptrf,  %rubout and ^O flag%                                       0878
      rubmrk= inptrf,                                                     0879
      inpstp, % ^S flag %                                                 0880
      rubabt,          %true if rubout causes abort%                      0881
      rubnocob=0,  %true to inhibit ^O (rubout) clearing output buffer%
                                                                         01122
   %...editing global variables...%                                       0882
      %...for new ring elements or data blocks...%                        0883
         dblst,   % index of last used data file block %                  0884
         rnglst,  % index of last used ring file block %                  0885
   %...set command and aptstr routine...%                                 0896
      modeoff=177B;  %  option set  %                                     0903
% sequence generator declarations %                                        024
   DECLARE EXTERNAL                                                      01124
   sqsavsw,   % global to save argument while switching stacks %         01125
   sqgwas[150],       % 15 sequence generator work areas %               01126
   sqgaend,                                                              01127
   sqsvws[130];       % 2 statement vector work areas; each is svmxlev +
   1 words long %                                                        01128
% TEMPS AVAILABLE TO ANY COMMAND %                                         030
   DECLARE EXTERNAL                                                      01055
      cm1tmp, cm2mp, cm3tmp, cm4tmp, cm5tmp, cm6tmp;                     01056
% jump name external file stid and name string %                         01100
   DECLARE EXTERNAL                                                      01010
                                                                         01011
```

```
      jfnefln= 0;                                                    01012
   DECLARE EXTERNAL STRING                                           01014
      ojnestr[200];                                                  01015
DECLARE EXTERNAL                                                      0126
   msstcount=0,                                                       0128
   mstxcount=0,                                                       0129
   msflag=0;  %indicates whether use measurements being recorded%     01083
% display core block requirements    %                               0133
% Declarations for NLS submodes%                                     0136
   % NLS utility subsystem %                                         01074
      DECLARE EXTERNAL                                               01075
         tskerrcnt;     % cnt of errors for run tasks %              01076
   % user program stuff %                                            0137
      DECLARE EXTERNAL                                               0138
         upgbuf= 554000B,   % start of user program buffer %         0139
         upgffbuf= 554000B,  % first free cell in upgbuf %           0143
         upgbend= 561777B,  % current end of user program buffer %   0140
         upgbsz= upgbdf,  % current size of user program buffer (in
         pages) %                                                    0141
         upgcacnt,  % number of Content analyzer patterns compiled %
                                                                     0142
         upgskix,   % index into upgstk %                           0144
         upgstk[upgsksz];   % stack of addresses of user programs % 0145
         DECLARE EXTERNAL STRING                                     0146
            upgnms[200];   % string to hold names of user programs %
                                                                     0147
   % run tenex subsystem stuff %                                     0990
      DECLARE EXTERNAL                                               0991
         tiw= 0,         % saved terminal interrupt word for this fork %
                                                                     0992
         tmode= 0,       % saved terminal mode word %                0993
         trmcod= 0,      % termination char terminal code %          0994
         ojfn= 0,        % -1 or output file jfn %                   0995
         ijfn= 0,        % -1 or input file jfn %                    0996
         pjfn= 0,        % jfn for the program %                     0997
         infork= 0;      % fork handle of inferior tenex subsystem % 0998
   % CALCULATOR strings %                                            0175
      DECLARE EXTERNAL STRING                                        0176
         opstring[30],   %char represetation, current operand%       0177
         acstring[30],   %char representation, current accum%        0178
         signstr[5],   %number of accum used as operand%             0179
         tstrng[5],    %number ofaccum used as operand%              0180
         astrng[5];    %number of the accumulator being used%        0181
   %Journal%                                                         0182
      DECLARE EXTERNAL                                               0183
         interflag, %interrogate mode flag%                          0184
         tmplt, %template submission flag%                           0185
         sabtfg, %submit abort flag%                                 0186
         jworkstid, %stid of jwork file%                             0187
         jcatstid, %stid of jcat%                                    0188
         jrnlstid, %Journal message file%                            0189
         diststid, %distributionn file%                              0190
         ctlstid, %stid for on-line distributior control files%      0191
         flagjfn=0, % jfn of nls flag file <netsys>nlsflags.flags %
                                                                     01093
         jdebug= 1, %true for debugginh...files go under duvall%     0192
```

```
    rfcflg, %True if RFC number has been assignnd%                    0193
    jdirn, %for saving number of connected directory (journal)%
                                                                      0194
    idirn, %for saving number of connected directory (ident)%         0195
    nsdcktime,  %next (gtad) time to check if system going down%
                                                                      01030
    lavtabad, %getab table address for load average%                  01031
    jdno, % address of string with list of numbers to be delivered;
    zero if none %                                                    01052
    jdid, % address of string with list of idents whose mail is to
    be processed-- zero if none %                                     01053
    jdfi, % address of string with list of files to be processed--
    zero if none %                                                    01054
    idfcnt, %counter for opening and closing ident file%              01021
    idfno, %file number of ident-file or zero%                        01022
    setidfno, %set by selection routines to note that they set
    idfno; used to reset only if they did the loading%                01117
    ckiwheel=1, %true if should check for identwheels when open
    identfile%                                                        01024
    idcrec, %address of current record string%                        01023
    idcrtype, %current record type (indtyp, grptyp, or orgtyp)%
                                                                      01027
    idcident, %address of current record ident string%                01025
    nwrecflag= 0, %true if user is defining a new record for the
    ident-file%                                                       01026
    idmodflag, %true if user is allowed to modify the current
    record%                                                           01048
    idmodified, %true if user has modified the current ident
    record%                                                           01049
    oljmlav=2026B8, %max load average (in floating point) for
    running on line del%                                              0196
    oljsbn, %six bit journal system name%                             0197
    %Now timing variables%                                            0198
       jtimfg, %True if timing%                                       0199
       jt0, jt1, jt2, jt3, jt4, jt5, jt6, jt7, jt8, jt9, jt10,
       jt11, jt12, jt13, jt14, jt15, %for individual times..see
       jtime for meanings%                                            0200
    ojsqsw, %seq work area for net jounal delivery%                   0201
    jlogjfn=0, %jfn of log file (<journal>jlog.txt) while primary
    output is being diverted there%                                   01094
    jokludgeupdtfl=0, %flag for updtfl to use ":" iistead of ";" in
    origin statement link for oldnls journal delivery at office-1%
                                                                      01095
    jsavaccess; %for restoring file access after Journal has run%
                                                                      0202
DECLARE EXTERNAL TEXT POINTER                                         0203
    jwp1, %pointer used to locate header in jwork%                    0204
    tmpbug, %pointer to user journal template%                        0205
    jb1, %misc pointers needed by Journal%                            0206
    jb2;                                                              01101
DECLARE EXTERNAL STRING                                               0207
    ipassw[40], %for saving password of connected director
    (ident)y%                                                         0208
    jpassw[40], %for saving password of connected directory
    (journal)%                                                        0209
    rfcnum[5], %for saving off RFC number if one is assigned%         0210
```

```
      datesr[20], %for date/time of entry%                              0211
      jnamstr[50]; %for building file names used by Journal%            0212
%Identification system%                                                 0214
   DECLARE EXTERNAL STRING                                              0215
      identstr[200], %used for building new entries%                    0216
      idwork[50], %work area%                                           0217
      newid[50], %used in collection of new id's%                       0218
      idntdel= "?
      ,()/\+!#$%_=@;*:<>[] ",                                          01086
      cmntdl= ")", %terminators for comments in identlists%             0221
      idnamdel[5];  %contains delimiters for reading names %            0222
   DECLARE EXTERNAL STACK                                               0223
      jidstk[20, 2];                                                   01102
   DECLARE EXTERNAL                                                     0224
      identwheel;  %true if user is a priviledged Ident system user%
                                                                       01104
   DECLARE EXTERNAL                                                     0225
      jidsbot;                                                         01103
%Number system%                                                        0226
   DECLARE EXTERNAL                                                     0227
      numstid; %stid of number file%                                    0228
%Hard copy Journal distribution%                                       0229
   DECLARE EXTERNAL                                                     0230
      % hcdiststid, Sometimes used for Global stid into hcdistfile%
                                                                        0231
      lptjfn, % Line ptr JFN--used globally by printing machinery%
                                                                        0232
      lptused, %Used as a count of number of documents which have
      been printed since the line printer was opened%                   0233
      docjfn, %JFN of Document portion of document being printed%
                                                                        0234
      hdrjfn, %JFN of header portion of document being printed%         0235
      docstrt, %Character displacement into docjfn file to bypass
      header%                                                           0236
      pfilnum, %file number used to tell printing machiinery when to
      change the document file%                                         0237
      lastfnum, %Used in conjunction winth lastfnum%                    0238
      prntfg, %True if anything has been priinted this running of
      hard copy distribution%                                           0239
      mastacprintflag, %print master and access copies on this
      machine%                                                         01029
      idfile; %Occaisionally used globally for file number of ident
      file%                                                             0240
   DECLARE EXTERNAL STRING                                              0241
      oprocn[50], %Contains name of Output Processor%                   0242
      ptstr[50], %Name of printer file--LPT or other%                   0243
      opstr[50], %Contains ident of operator%                           0244
      docwfn= "<journal>DPNTWRK.txt.0
      ", %Name of document print work file%                             0245
      hdrwfn= "<journal>HPNTWRK.txt.0
      "; %Name of header print work file%                               0246
%FTP routines%                                                         0247
   DECLARE EXTERNAL                                                     0843
      ftphnd= 0,                                                       01105
      ftpem,                                                           01109
      ftperm,                                                          01108
```

```
        ftpoh,                                                          01107
        ftpoha;                                                         01106
    %collector sorter%                                                   0250
      DECLARE EXTERNAL                                                  0251
        keypr,                                                          0252
        vcvulp,                                                         0253
        vcvtp,                                                          0254
        mersiz,                                                         0255
        merrff,                                                         0256
        vtop, %vector index%                                            0257
        infopn, %flag--true if input file open%                         0258
        cstid, %stid of output file%                                    0259
        outcnt, %count of statements in current output file%            0260
        smtmax= 10000, %maximum numbr of statements in one output file%
                                                                        0261
        colda, %address of display area%                                0262
        colsw, %addess of sequence area%                                0263
        sortfg, %true if we are to sort%                                0264
        namndx, %index into input file name list%                       0266
        namlst[50], %pointers to text for input file names (indexed by
        naamndx%                                                        0267
        nambuf[100], %text area for input file names%                   0268
        ninfil, %number of input files%                                 0269
        strpkey, %true if delete keys%                                  0270
        cversion; %Number in sequence of output file%                   0271
      DECLARE EXTERNAL STRING                                           0272
        outnam[50]; %contains root name for output files%               01110
    %Catalog system%                                                    0273
      DECLARE EXTERNAL                                                  0274
        cppflag= FALSE; % Catalog Production Processor in Operation
        Flag %                                                          0275
% DECLARE's %                                                          0276
  DECLARE EXTERNAL                                                      0277
    exp=0,      %experimental system flag%                              0279
    hostni= -1, % LH is HOSTN table #; RH is # entries in HOSTN %       0818
    hostsi= -1,  % LH is HSTNAM table #; RH is # entries in HSTNAM %
                                                                        0819
    ttytbl,    % system JOBTTY table number %                         01000
    jobtbl,    % system JOBDIR table number %                         01001
    bndchk,    % TRUE=> perform boundary checks at tt (qt) %          01079
    lstred,    % last read date of initial file for user programs %
                                                                        01070
    acchecked,          %flag:  access to the file was checked%      01119
    skpachk= FALSE,     %skip access check flag -- used?? %          01120
    ignqrps= 0,         %address of a string to hold login groups%   01121
  %....CALCULATOR.....%                                                 0309
    %accumulators %                                                     0310
      accum[20],                                                        0311
      asub,      %subscript of current accumulator%                   0312
      vaccum[2],  %scratch accum for EVALUATE%                         0313
      opfloat[2], %double floating current operand%                    0314
      chadent= 0, %TRUE if user enters calc from the calc %            0823
      nofeedbk,   % true if TNLS user set terse feedback %             0315
    % misc %                                                           0316
      cda,       %display area address, may be actual or pseudo%      0317
      castid,    %current location in CALC-IDENT file%                0318
```

```
      proc,    %addr of arithmetic execution routine%                    0319
   % format variables %                                                  0320
      cadflg= 0,                                                         0321
      calflg= 0,                                                         0322
      cacflg= 0,                                                         0323
      dfoutm= 064014120200B; % mask for dfout JSYS %                     0324
DECLARE EXTERNAL STRING                                                  0344
   vrsnno[5],  %save area for version number%                            0352
   ladj[40],   %string forlevel adjust%                                  0360
   sfhead[100],                                                          0361
   sfstng[100],                                                          0362
   lkupreg[100], %<JUMP, lookup> string save area%                       0370
   prbuf[150];  %typewriter output buffer%                               0371
DECLARE EXTERNAL TEXT POINTER                                            0396
   p3,  p4,  p5,  p6,  p7,  p8,  p9,  p10;                               0397
DECLARE EXTERNAL                                                         0400
%...file status table...%                                                0406
   filst[100],  %1 entry per file, 25 (filmax) files,  4 (filstl)
   words per entry%                                                      0407
   filcnt= 0,                                                            0408
%...substitute work area...%                                            0495
   subrec[subnm1],                                                       0496
   subhed[4], %=lshed -- see (UTILTY, shed)%                             0497
%...file access stuff...%                                               0501
   access= 1, %normal access%                                            0502
   accmask= (1, 2, 4, 10B, 20B, 40B, 100B, 200B),                        0503
   normlaccess= 0,                                                       0504
   jrnlaccess= 1,                                                        0505
   debugaccess= 1,                                                       0506
%...file header...%                                                     0507
   % DONT CHANGE THE ITEMS IN THE HEADER %                              0508
   filhed[5],                                                            0509
      % these words now used for link def. directory string%            0510
      %see fust0%                                                      01129
   fcredt,   % file creation date %                                      0511
   nlsvwd= 1,   % nls version word (keyword) %                           0512
   sidcnt,   %count for generating SID's%                                0513
   finit,    % initials at last write %                                  0514
   funo,     % user number (file owner) %                                0515
      %if <0, RH is pointer to string in fileheader%                    0844
   lwtim,    % last write time %                                         0516
   namdl1,   % left name delimiter default character %                   0517
   namdl2,   % right name delimiter default character %                  0518
   rngl,     % upper bound on data ring file blocks %                    0519
   dtbl,     % upper bound on data file blocks %                         0520
   rfbs[6],    % start of random file block status tables %             0521
   rngst[95],  % ring block status table %                              0522
   dtbst[370], % data block status table %                              0523
   mkrtxn= 20,   % marker table maximum length %                        0524
   mkrtbl,     % number of markers in marker table %                    0525
      %each marker takes MKRL words%                                    0845
   mkrtb[20],  % marker table %                                         0526
   filhde,   %end of the file header%                                   0527
%...page 1 file header addition...%                                    01130
   filhd1[16], %link default string when too big for page 0%           01131
      %see fust1 (and fust0)%                                          01132
```

```
%...output sequential/quickprint...%                                    0528
   sfpmr1,  %r1 for pmaps to output file%                               0529
   sfpmr2,  %r2 for pmaps to output file%                               0530
   sfpmr3,  %r3 for pmaps to output file%                               0531
   sfbyte,  %number of bytes in file%                                   0532
   sfbufl,  %length of output buffer ( in bytes)%                       0533
   sfucf,   %if true, convert to upper case%                            0534
   sfpgno,  %if true, paginate; contains current page number%           0535
   sflnel,  %number columns (characters) per line%                      0536
   sfndlv,  %number characters per level to indent%                     0537
   sfmxnd,  %maximum number characters to indent%                       0538
   sfndbf,  %bytptr to end of op buffer%                                0539
   sfbptr,  %byte pointer to current poosition in buffer%               0540
   sflnpg,  %number of line, this page%                                 0541
   oqnhfg,   %put only page # on output quickprint if true%            01087
   oqapfg,  %try open append for D Q if true%                           0542
%...insert sequential work area...%                                    01091
   levind= (0,0,0,0,0,0,0,0,0,0,0,0),                                  01092
%...High segment page map table -- see (seqfil, processor)%            0543
   freemap[freesz],                                                     0544
   pmfjfn=0,  %jfn of "pmf" file for private pages of high seg when
   mapped out%                                                         01123
%...other work areas...%                                               0545
   mswsit, %cell to accumulate time for wsi%                            0547
   mswsic, %cell to accumulate count for wsi%                           0548
   %wsd-meas cells%                                                     0549
      ldrfct, %Number of calls on lodrfb%                               0550
      ldrnct, %number of calls on lodrng%                               0551
      ldsdct, %Number of calls on lodsdb%                               0552
      fchsct, %numbr of calls on fchsdb%                                0553
      strttm, %JOBTM at start of NLS (intmeas)%                         0554
   msaddr[15],  %for measurement -- 3 * number of entries%              0555
   mslst[39],  %for measurement -- msentl * number of entries%          0556
   msiste,  %end of measurement list%                                   0557
   msr1sv,  %save area for r1 during measurement%                       0558
   msr2sv,  %save area for r2 during measurement%                       0559
   msr3sv,  %save area for r3 during measurement%                       0560
   measnt,  %current item being measured%                               0561
   iswork[4],   %insert sequential work area%                           0562
   oprwrk[20],  %output processor work area%                            0563
   comexflag= 0, %KLUDGE until output COM syntax fixed-- 0 is comp80,
   1 is singer%                                                        01098
   gtjfnt[10],  %for long getjfn calls%                                 0564
   typend,                                                             01081
%...global display/print parameters...%                                0567
   %...display global support...%                                       0568
      lastch,                                                           0574
      % mkrbuf,          pointer to marker display buffer%             0575
      mkrend,          %ditto%                                          0576
      mkrptr,          %pointer into marker display buffer%             0577
      mkrcnt,          %char count of last marker found%                0578
      mkrflg,          %true if found a marker in a statement%          0579
      % cdctrn, %                                                       0580
      cdchr1,                                                           0581
      cdchct,                                                           0582
   %...fast create display support...%                                  0590
```

```
      aulsrt,  %address, current auxiliary display table%          0591
      aulsize,  %size, in lsrtl entries, of auxiliary table%       0592
      rttop,   %starting address, current table being formatted%   0593
      rtsize,    %size, current table being formatted%             0594
   %...display support...%                                        0595
      % aplint, %                                                 0596
      udpnwrap,  %number of wraps so far in this line%           01099
      gapcol,                                                     0597
      gapcnt,                                                     0598
      gapbp,                                                      0599
      gapcc,                                                      0600
      cc,                                                         0601
      rt,                                                         0602
      rte,                                                        0603
      art,                                                        0604
      arte,                                                       0605
      oldlsrt,  %address of old Line Segment Reference Table%     0606
      commands[65], %first block of display commands--others to be
      dynamically allocated%                                      0610
      dpcmblk, %address of commands block currently being filled%
                                                                  0611
      dgstl,    %current character count for statement formatting%
                                                                  0612
      dgstml,   %max character count%                             0613
      dgstid,    %stid of statement being formatted%              0614
      dgbufe;                                                     0615
%...typewriter print variables...%                                0621
   DECLARE EXTERNAL                                               0622
   prmkrf= 0,    %typewriter print marker flag%                   0623
   cdpagf= 1,    %page flag%                                      0624
   pageno,    %page number for print group%                       0629
   slshfg,    %slash command flag%                                0632
%...general global variables...%                                 0633
   bfileno, %Contains file number on bad file SIGNAL%             0636
   ercall, %Contains the address of last call to error%           0637
   ermark, %Contains value of mark at last call to err%           0638
   fastname= 1, %Flag for which jump name algorithm to use on normal
   jumps%                                                         0639
   carpos,        %position of carraige (0=left margin)%          0647
   slashflg= 0,  %show selections flag%                           0652
   srchtype, %Type of item to search for (name/word) contents%    0656
%...Library  stuff...%                                           0664
   oldflg= 0, %sticky state of libflg%                            0842
   libflg= 0, %non-zero means routine running using NLS as library
   (so don't expect input)...number indicates which%              0665
   liblod= 0, % TRUE if load average cut-off set by hand %       01050
   jnlprog= 0, % TRUE if journal userprog in; FALSE if not. %    01051
   autostrt= 0, %True if the library started automatically at system
   startup%                                                       0666
   slnkrv= 1,                                                     0668
   utiltv= 4,                                                     0669
%...processor variables...%                                      0675
   prseqwk,  %address of  processor's seqgen work area%           0676
   opbp,     % byte pointer to start of current string %          01004
   opcbp,    % byte pointer to current char position %            01005
   opccpos,  % current char postion %                            01006
```

```
      opcmax,   % max chars in current string %                          01007
      oplev,    % current statement level %                             01008
      opnewst,  % flag, used to make level find efficient %             01009
   %...substitute variables...%                                          0677
      subcnt;     %number of substitutions made%                         0678
   %...Print Journal...%                                                 0780
      DECLARE EXTERNAL                                                   0781
         pjrbab, % save for contents of rubabt %                         0782
         pjusqc, % save for pntr to user seq gen %                       0783
         pjsavf, % save for user seq gen viewspec %                      0784
         pjsw,                                                           0785
         pjlsw, % pntrs to seq workareas %                              01116
         pjrubout; % set TRUE on RUBOUT %                                0786
      DECLARE EXTERNAL TEXT POINTER                                      0787
         pjstid;                                                        01111
                                                                         0788
   DECLARE EXTERNAL                                                      0801
      patch[40], % patch space %                                        01112
      patche;                                                           01113
                                                                        0802
% page aligned output buffers %                                         0846
   PAGE 140B; %get aligned on page boundary (assume load starts 140B)%
                                                                         0847
   DECLARE EXTERNAL                                                      0848
      msbuff[511],  %measurement file buffer%                           01114
      msbufe;  %end of measurement buffer%                               0849
   DECLARE EXTERNAL                                                      0850
      sfbuff[511],  %Sequential file buffer%                            01115
      sfbufe;  %end of Sequential file buffer%                           0851
FINISH of bdata                                                         0817
```

BINTNLS

```
< NLS, BINTNLS.NLS.15, >, 23-Apr-78 10:35 JDH ;;;;
FILE bintnls  %  L10 to <rel-nls>BINTNLS %% (L10,) (rel-nls,BINTNLS,) %
%......Declarations......%
   REF rawchr;
   REGISTER
      a1 = 12, %working register%
      a4 = 15, %working register%
      r1 = 1,  %working register%
      r2 = 2,  %working register%
      r3 = 3,  %working register%
      r4 = 4,  %working register%
      r5 = 5,  %working register%
      rp = 11, %record pointer%
      p = 7,   %pattern stack%
      s = 9,   %general call stack pointer%
      m = 10;  %mark stack pointer%
   EXTERNAL binit, bqt;
%.....Entry points.....%
   (dex) PROCEDURE; %start DEX%
      !JSP r1,l10init;
      initback($dxin);
      LOOP !haltf;
      END.
   (dxin)PROCEDURE; % Roll in dxctl file %
      LOCAL proc, result[10]; % Address of DXSTRT procedure in the user
      program. %
      LOCAL TEXT POINTER tp;
      LOCAL STRING dxname[20];
      REF proc;
      % Roll in DEX user program. %
         % Increase buffer size to 30 pages. %
            IF NOT gpbsz( 30 ) THEN
               BEGIN
               btypclstr($"Cannot set program buffer for DEX system!");
               LOOP !haltf;
               END;
         % Reset the user program buffer. %
            gpgmrst();
         % Load the user program with DEX code. %
            *dxname* _ "rel-nls, dxctl";
            ON SIGNAL ELSE
               BEGIN
               btypclstr($"Cannot load DEX program");
               LOOP !haltf;
               END;
            getuprog($dxname);
            ON SIGNAL ELSE;
         % Get address of DXSTRT %
            IF NOT ddtlookup($"DXSTRT", FALSE, % get procedure in user
            program. %: &proc) THEN
               BEGIN
               btypclstr($"Cannot find entry procedure");
               LOOP !haltf;
               END;
      % Run DEX. %
         proc();
```

```
   btypclstr($"Completed.");
   LOOP !haltf;
   END.
(quin)PROCEDURE(nlsparse); % Roll in query file %
   LOCAL proc, result[10]; % Address of QPORT procedure in the user
   program. %
   LOCAL TEXT POINTER tp;
   LOCAL STRING quname[20];
   REF proc;
   % Roll in QUERY user program. %
      % display message to user %
         btypclstr($"Loading Query System");
      % Reset the user program buffer. %
         gpgmrst();
      % Load the user program with QUERY code. %
         *quname* _ "programs, query";
         ON SIGNAL ELSE
            BEGIN
            btypclstr($"Cannot load QUERY program");
            LOOP !haltf;
            END;
         getuprog($quname);
         ON SIGNAL ELSE;
      % Get address of QSUPER %
         IF NOT ddtlookup($"QSUPER", FALSE, % get procedure in user
         program. %: &proc) THEN
            BEGIN
            btypclstr($"Cannot find query entry procedure");
            LOOP !haltf;
            END;
   % Run QUERY. %
      proc(nlsparse);
   btypclstr($"Leaving Query.");
   LOOP !haltf;
   END.
(jbackground) PROCEDURE; %start Journal background process%
   !JSP r1,l10init;
   autostrt _ 1;
   initback($jnlrun);
   LOOP !haltf;
   END.
(jnlrun)PROCEDURE; % Run journal delivery via JOUTIL %
   LOCAL proc; % Address of JOUTIL procedure in the user program. %
   REF proc;
   jnlin(); % get JNLDEL file%
   % Get address of joutil %
      IF NOT ddtlookup($"JOUTIL", FALSE, % get procedure in user
      program. %: &proc) THEN
         BEGIN
         btypclstr($"No JOUTIL procedure:  error in DDT lookup");
         LOOP !haltf;
         END;
   IF libflg THEN % run journal delivery %
      BEGIN
      proc( libflg _ libflg .V oldflg, jdid % list of idents. %);
      btypclstr($"Completed.");
```

```
      END
   ELSE btypclstr($"No library functions set");
   RETURN;
   END.
(jnlin)PROCEDURE; % Roll in JNLDEL file %
   LOCAL STRING jname[20];
   IF jnlprog THEN RETURN(FALSE); %already in%
   % Take care of interrupts if this is autostart %
      IF autostrt THEN setinterrupts();
   % Roll in Journal user program. %
      % Increase buffer size to 30 pages. %
         IF NOT gpbsz( 30 ) THEN
            BEGIN
            btypclstr($"Cannot set program buffer for JOurnal
            system!");
            LOOP !haltf;
            END;
      % Reset the user program buffer. %
         gpgmrst();
      % Load the user program with Journal delivery/utility code. %
         *jname* _ "netsys, jnldel";
         ON SIGNAL ELSE
            BEGIN
            btypclstr($"Cannot load jnldel user program");
            LOOP !haltf;
            END;
         getuprog($jname); % loads user program, does  not put out
         messages. %
         ON SIGNAL ELSE;
      % Set flag indicating that user program has been rolled in. %
         jnlprog _ TRUE;
   RETURN(TRUE);
   END.
(btypestr) PROCEDURE (astrng); %Type a-string on tty%
   %-----------------%
   REF astrng;
   IF NOT astrng.L THEN RETURN;
   !sout(101B, chbmty + &astrng, -astrng.L);
   RETURN;
   END.

(btypclstr) PROCEDURE (astrng); %type CRLF followed by astrng%
   LOCAL STRING crlfstr[5];
   *crlfstr* _ CR, LF;
   btypestr($crlfstr);
   btypestr(astrng);
   RETURN;
   END.
(net) PROCEDURE; % Start NLS from the Net %
   r1.LH _ 7; % first AC to save is 7 %
   r1.RH _ $xacs [7]; % that's where it's to be saved %
   r2 _ $xacs [15]; % AC 17 is last one to be saved %
   ! BLT 1,(2); % save ACs 7-17 %
   !JSP r1,110init;
   initback($netdsp);
   LOOP !haltf;
```

```
    END.
DECLARE submission=1;
(netdsp) PROCEDURE; % Dispatch to HIOP or NJS %
    % Declarations %
        LOCAL reason;   % reason for dispatch %
    % Save dispatch reason %
        r1 _ xacs [7]; % get xwd dispatch-reason, xxx %
        ! HLRZS 1; % clean dispatch reason %
        reason _ r1; % save it %
        IF reason = submission THEN njs($xacs) ELSE hiop($xacs);
        END.
%.....NLS pre-initialization and SSAVE code .....%
(bpresaveinit) PROCEDURE; %partial initialization of backend before
SSAVEing it as an object file %
    LOCAL return, char;
    IF FALSE THEN
        BEGIN
        (bqt): !JSP r1,l10init;
        return _ FALSE;
        END
    ELSE return _ TRUE;
    ON SIGNAL ELSE
        BEGIN
        IF sysmsg AND [sysmsg].L <= [sysmsg].M AND [sysmsg].L IN [1,
        200] THEN
            typeas(sysmsg);
        !JSYS 147B; %Reset jsys-- cannot use symbol because of
        conflict%
         LOOP !haltf;
            %In case someone is foolish enough to try to contnue%
        END;
    IF jdebug THEN % check journal stuff %
        LOOP
            BEGIN
            typeas($"Enable Journal System?? ");
            CASE rawchr() OF
               = 'y, = 'Y: %yes%
                  BEGIN
                  jdebug _ 0;
                  typeas($"es.   Password for Directory Journal (CR for
                  default) = ");
                  *jnlpsw* _ NULL;
                  LOOP
                     BEGIN
                     IF (char_rawchr()) = EOL THEN EXIT;
                     *jnlpsw* _ *jnlpsw*, char;
                     END;
                  IF jnlpsw.L = 0 THEN *jnlpsw* _ "JPD";
                  EXIT LOOP;
                  END;
               = 'n, = 'N: %no%
                  BEGIN
                  typeas($"o.");
                  *jnlpsw* _ "JPD";
                  EXIT LOOP;
                  END;
```

```
            ENDCASE
               BEGIN
               typeas($" ? (y or n)");
               crlf();
               REPEAT LOOP;
               END;
         END;
      IF hdebug THEN % check help stuff %
         LOOP
            BEGIN
            typeas($"Enable Help System?? ");
            CASE rawchr() OF
               = 'y, = 'Y: %yes%
                  BEGIN
                  hdebug _ 0;
                  typeas($"es.");
                  crlf();
                  EXIT LOOP;
                  END;
               = 'n, = 'N: %no%
                  BEGIN
                  typeas($"o.");
                  crlf();
                  EXIT LOOP;
                  END;
               ENDCASE
                  BEGIN
                  typeas($" ? (y or n)");
                  crlf();
                  REPEAT LOOP;
                  END;
            END;
      % Set entry vector %
         setbvec();
      % initialize free storage zone %
         % LATER this will have its own free storage zone -- noop for
         now
         makezone($dspblk, 2000B, 8, $dspbke - $dspblk-1);
         %
      % initialize sequence work areas %
         sqinit();
      % initialize file stuff %
         filinit();
      % initialize user program stuff %
         gpgmrst();
      % initialize strings for ident system%
         *idnamdel* _ '?, CR, EOL;
      IF return THEN RETURN
      ELSE
         BEGIN
         backsave();
         LOOP !haltf;
         END;
      END.
%.....NLS initialization .....%
   (initback) PROCEDURE (dispatcher); %start up the backend%
```

```
    % This is the procedure which is called in order to start up
    NLS-backend%
    %------------------%
    %!!!CANNOT HAVE LOCAL VARIABLES!!!%
    IF FALSE THEN
       BEGIN
       (binit): %label inserted to avoid the first instruction in the
       procedure which verifies that the stack pointer is reasonable%
       !JSP r1, l10init;  %initialize l10 runtime environment%
       dispatcher _ 0;
       END;
    % go start the world %
       startback(dispatcher);
    LOOP !haltf;
    END.

(startback) PROCEDURE (dispatcher); %initialize the backend%
    %-----------------%
    % initialization done every startup or re-startup %
       ON SIGNAL ELSE % Close files and issue halt jsys if error in
       startup %
          BEGIN
          IF sysmsg AND [sysmsg].M AND [sysmsg].L AND [sysmsg].M >=
          [sysmsg].L AND [sysmsg].L < 200 THEN typeas(sysmsg);
          (strthlt):
             !JSYS 147B; %Reset jsys-- cannot use symbol because of
             conflict%
              LOOP !haltf;
                %In case someone is foolish enough to try to contnue%
          END;
       %Check to see if we are to run library routines%
          libflg _ IF libflg THEN libflg ELSE checklib();
          IF libflg = 1 THEN
             BEGIN
             setpriority( 202B);
             libflg _ 11B;
             END;
       % misc. stuff %
          inptrf _ FALSE; %rubout flag%
          lhostn _ hostnumber(); %save logical host number%
          %misc. for background, load averag checks%
             nsdcktime _ 0; %time (gtad) to check if system going
             down%
             !sysgt(getsbn($"SYSTAT"));
             !HRLI r2,14B; lavtabad _ r2; %getab table address of load
             average%
          tenex _ tenexverno();
    IF NOT continue THEN % this stuff done only once %
       BEGIN
       % initialize measurement tables %
          IF msflag THEN intmeas();
       % initialize globals about monitor tables %
          !sysgt(getsbn($"JOBTTY"));
          ttytbl _ r2.RH;  % table number %
          !sysgt(getsbn($"JOBDIR"));
          jobtbl _ r2.RH;  % table number %
```

```
      % create message fork %
         % will need this LATER, but noop now
         IF nlmode = fulldisplay THEN inittimer();
         %
      %get mode, system name%
         *nlsnam* _ "BACKNLS";
         nlstyp _ tntyp;
      END;
    IF dispatcher THEN [dispatcher]() ELSE RETURN;
   END.

%.....Initialization support routines.....%
   (backsave) PROCEDURE; % save nls as disk file %
      LOCAL jfn;
      LOOP % get save file name from user %
         BEGIN
         typeas($"save filename: ");
         IF NOT SKIP !gtjfn(400003B6,100000101B) THEN
            BEGIN
            typeas($" ?");
            crlf();
            REPEAT LOOP;
            END;
         jfn _ r1;
         EXIT LOOP;
         END;
      % save core image as dsk file %
         !ssave(400000B6 .V jfn.RH, 777000B6 .V 520000B, 0);
            %WILL HAVE TO CHANGE THIS LATER FOR CORRECT BOUNDS%
      RETURN;
      END.

   (setbvec)PROCEDURE; %set entry vector for backend%
      % comment out for now
         entvec[0] _ $binit .V 254B9; start backend%
      entvec[3] _ $dex .V 254B9; %start DEX%
      entvec[4] _ $jbackground .V 254B9; %journal background process%
      entvec[6] _ $net [1] .V 254B9; %start from Network %
      !sevec( 4B5, 10B6 .V $entvec); %set entry vector%
      RETURN;
      END.
   (filinit) PROCEDURE;  %init file handling mechanisms%
      LOCAL
         fileno, count, fz, da, astrng, fl, fp, end, hdr, sndptr, char;
      REF sndptr, fl, fp, astrng, fz, da;
      %initialize file list%
         nxpg _ 1;  %for lodrfb%
         &fl _ $filst;
         end _ &fl + filmax*filstl;
         DO fl _ 0 UNTIL (&fl _ &fl + 1) = end;
         &fl _ $filst;
      %Initialize filepart table%
         &fp _ $filepart;
         end _ &fp + filmax;
         DO fp _ 0 UNTIL (&fp_&fp+1)=end;
      %initialize frozen list%
```

```
      &fz _ fzfree _ $fzlsts;
      end _ &fz + (frzmax-1)*frzl;
      DO
         fz.fznext _ &fz + frzl
      UNTIL (&fz _ &fz + frzl) = end;
      fz.fznext _ 0;
   RETURN END.

(checklib)PROC;
   %This procedure checks the system flags to see if it should try to
   start up one of the background processors automatically.
      It so, it tries to start it up.
      If successful, it returns the library number for the processor,
      .
      OtherWise, it may call err, or it may return 0 depending on the
      error%
   LOCAL libndx;
   IF nojournalflag THEN RETURN(0);
   libndx _ 0;
   IF libstrt(2) THEN RETURN(1); %will start as auto job if not
   logged in%
   %Make sure this was not a false start%
      !gjinf;
      IF r1 = 0 THEN
         killib(); %Not logged in--kill job%
   RETURN(0);
   END.

DECLARE libsw1=4625320474B2, libsw2=51607014IB3;
(libstrt)PROCEDURE(libno);
   LOCAL libdirno, temp;
   LOCAL STRING errstr[200];
   %This procedure tries to login and start a background processor.
   It may do a directory connect too depending on the processor%
   %Set up to catch signals%
      ON SIGNAL ELSE
         BEGIN
         IF NOT sysmsg THEN sysmsg _ $"Error";
         *errstr* _ "Library Start Fail: ", *[sysmsg]*;
         specttyout(0,$errstr);
         specttyout(oprtty, $errstr);
         killib();
         END;
   %see if job is logged in%
      IF (libdirno_!gjinf()) # 0 AND NOT autostrt THEN RETURN(FALSE);
   %Set "XXX" ident%
      *initsr* _ "XXX";
      cinit _ 3061400B; %could use setcinit, but it is in the
      frontend%
      nwheelf _ 1;
   %Get dirno of user for login%
      IF NOT (temp _ stusrcall($"JDEMON", 0)) THEN specttyout(0,
      $"Illegal Library User");
   %Handle "already logged in"%
      IF libdirno  # 0 THEN
         BEGIN
```

```
        RETURN(IF temp = libdirno THEN TRUE ELSE FALSE);
        END
      ELSE libdirno _ temp;
   %enable login capability%
      enablLogin();
   %log him in%
      r1 _ r1.RH;
      r2 _ 4407B8 + (CASE lhostn OF
         =utilhost: $libsw1;
         ENDCASE $libsw2);
      IF NOT SKIP !login(r1, r2, 5B11 .V 1) THEN
         BEGIN
         *errstr* _ "Could Not Log In Background Job", CR, LF,
         STRING(r1);
         specttyout(0, $errstr);
         specttyout(oprtty, $errstr);
         killib();
         END;
   autostrt _ TRUE;
   %Make it so we are not auto-logged out%
      !atljb(1);
   RETURN(libno);
   END.

(killib)PROC;
   !lgout(-1); %log job out%
   END.

(enablLogin)PROC;
   %Enable the Login capability%
   !rpcap(4B5);
   IF NOT r2.bit3 THEN SIGNAL(0, $"Log Not in Assigned
   Capabilities");
   r3.bit3 _ 1;
   !epcap;
   RETURN;
   END.

%measurement stuff%
   SET
      begloc=0, endloc=1, begins=2, endins=3,
      measmx=4, meascr=5, frstclk=6, totclk=7;
   SET ZERO = 0;
   (intmeas) PROCEDURE;  %measurement initialization%
      LOCAL curitm, jsrbins, jsreins, count;
      REF curitm;
      %wsd-meas%
         !JSYS jobtm;
         strttm _ r1;
         ldrnct _ ldrfct _ ldsdct _ fchsct _ 0;
      &curitm _ $mslst;
      jsrbins _ 264B9 .V $begmeas;
      jsreins _ 264B9 .V $endmeas;
      count _ 0;
      DO
         IF (curitm.msbegin _ msaddr[count]) THEN
```

```
            BEGIN
            IF NOT (curitm.msend _ msaddr[count _ count + 1]) THEN
               BEGIN
               dismes (2, $"Measurement begin address has no
               corresponding end address");
               EXIT LOOP;
               END;
            curitm.msbinstn _ [curitm.msbegin] := jsrbins;
            curitm.mseinstn _ [curitm.msend] := jsreins;
            BUMP curitm.msbegin, curitm.msend, count;
            curitm.msmax _ msaddr[count := count + 1];
            curitm.mscurr _ curitm.ms1clk _ curitm.mstclk _ 0;
            END
         ELSE count _ count + 3
      UNTIL (&curitm _ &curitm + msentl) >= $mslste;
      RETURN;
      %the following assembly code is executed when the measurement
      routines are called%
         %this routine records the clock time to begin measurement%
            (begmeas): !ZERO;
            !MOVEM r1,msr1sv;  %save r1, r2, r3%
            !MOVEM r2,msr2sv;
            !MOVEM r3,msr3sv;
            !MOVEI r1,mslst;  %index to possible measurement entry%
            (bmsfnd): !MOVE r2,begloc(r1);
            !HRRZ r3,begmeas;
            !CAMN r2,r3;
            !JRST bmstim;  %this is the entry%
            !ADD r1,msentl;
            !CAIGE r1,mslste;
            !JRST bmsfnd;
            !MOVE r1,=255B9;  %entry not here -- fake return%
            !MOVEM r1,bmsrin;
            !JRST bmsrtn;
            (bmstim): !MOVEM r1,measnt;
            !MOVE r1,=-5;
            !JSYS jobtm;
            !MOVE r3,measnt;
            !MOVEM r1,frstclk(r3);
            !MOVE r2,begins(r3);
            !MOVEM r2,bmsrin;
            (bmsrtn): !MOVE r1,begmeas;
            !HRRM r1,bmsrloc;
            !MOVE r1,msr1sv;
            !MOVE r2,msr2sv;
            !MOVE r3,msr3sv;
            (bmsrin): !ZERO;
            (bmsrloc): !JRST 0;
         %this routine records the clock time and performs the
         necessary calculations at the end of the measurement
         interval; to read the measurement at the end of the
         specified number of passes, two breakpoints may be set
            at msmbrk: at this point r1 will contain the elapsed time
            in milliseconds;
            at mssbrk: at this point r1 will contain the elapsed time
            in seconds, and r2 the remainder, in milliseconds%
```

```
            (endmeas): !ZERO;
            !MOVEM r1,msr1sv;  %save r1, r2, r3%
            !MOVEM r2,msr2sv;
            !MOVEM r3,msr3sv;
            !MOVEI r1,mslst;  %index to possible measurement entry%
            (emsfnd): !MOVE r2,endloc(r1);
            !HRRZ r3,endmeas;
            !CAMN r2,r3;
            !JRST emstim;  %this is the entry%
            !ADD r1,msentl;
            !CAIGE r1,mslste;
            !JRST emsfnd;
            !MOVE r1,=255B9;  %entry not here -- fake return%
            !MOVEM r1,emsrin;
            !JRST emsrtn;
            (emstim): !MOVEM r1,measnt;
            !MOVE r1,=-5;
            !JSYS jobtm;
            !MOVE r3,measnt;
            !SUB r1,frstclk(r3);
            !ADDM r1,totclk(r3);
            !AOS 1,measCr(r3);
            !CAMGE r1,measmx(r3);
            !JRST emscon;
            !MOVE 1,totclk(r3);
            (msmbrk): !IDIVI r1,1000;
            (mssbrk): !SETZM measCr(r3);
            !SETZM totclk(r3);
            (emscon): !MOVE r2,endins(r3);
            !MOVEM r2,emsrin;
            (emsrtn): !MOVE r1,endmeas;
            !HRRM r1,emsrloc;
            !MOVE r1,msr1sv;
            !MOVE r2,msr2sv;
            !MOVE r3,msr3sv;
            (emsrin): !ZERO;
            (emsrloc): !JRST 0;
        END.


FINISH
This code can be deleted after 1-august-74.    -- Charles
   (oldintdspda) PROCEDURE (tat, tab, tal, tar);  %set up display areas
   for display terminal (relative to position of text area)%
      %sets up top six lines as control and literal feedback areas%
      %text area%
         tatop _ tat;
         tabottom _ tab;
         taleft _ tal;
         taright _ tar;
      %lt viewspec%
         lttop _ tat - 4*tavinc;
         ltbottom _ lttop + ltvinc;
         ltleft _ tal;
         ltright _ ltleft + 7*lthinc;
      %viewspec%
```

```
      vstop _ tat - 3*tavinc;
      vsbottom _ vstop + vsvinc;
      vsleft _ tal;
      vsright _ vsleft + 7*vshinc;
   %name%
      namtop _ tat - 4*tavinc;
      nambottom _ namtop + namvinc;
      namright _ tar;
      namleft _ tar - 25*namhinc;
   %sybsystem name%
      subtop _ tat - 3*tavinc;
      subbottom _ subtop + subvinc;
      subright _ tar;
      subleft _ subright - 25*subhinc;
   %Command Feedback Line%
      cfltop _ tat - 4*tavinc;
      cflbottom _ cfltop + 2*cflvinc;
      cflleft _ ltright + cflhinc;
      cflright _ namleft - cflhinc;
   %literal feedback%
      littop _ tatop - 2*tavinc;
      litbottom _ tabottom;
      litleft _ taleft;
      litright _ taright;
   %message (tty) area%
      msgtop _ tat - 6*tavinc;
      msgbottom _ msgtop + 2*tavinc;
      msgleft _ tal;
      msgright _ tar;
   RETURN;
   END.
```

RECORDS

```
< NLS, BRECORDS.NLS.8, >, 21-Apr-78 14:17 JDH ;;;;
FILE brecords % L10  <REL-NLS>brecords %% (L10,) (rel-nls,brecords.rel,)
%
% records for use with the directory commands %
   (fdbprr) RECORD  % description of fdbprt word in fdb %
      flprpu[6],        % public protection bits %
      flprgr[6],        % group protection bits %
      flprse[6];        % self protection bits %
   (fdbprb) RECORD  % description of protection field bits %
      flprnu[1],        % not used %
      flprli[1],        % list protection %
      flprap[1],        % append access %
      flprex[1],        % execute access %
      flprwr[1],        % write access %
      flprre[1];        % read access %
   (fdbctr) RECORD  % description of fdbctl word in fdb %
      fdbctu[18],       % unused bits %
      fdbeph[1],        % file is ephemeral %
      fdbnun[11],       % more unused bits %
      fdbing[1],        % this is a long file %
      fdbnxf[1],        % file doesn't exist yet - unused %
      fdbdel[1],        % file is deleted %
      fdbnex[1],        % no ext for this fdb yet - unused %
      fdbprm[1],        % file is permanent %
      fdbtmp[1];        % file is temporary %
   (fdbarr) RECORD  % description of fdbart word in fdb %
      flsnat[18],       % second archive tape number %
      flfsat[18];       % first archive tape number %
   (fdbbfr) RECORD  % description of fdbbkf word in fdb %
      fldmpt[18],       % most recent dump tape number %
      flarfl[18];       % archive flags  (see fdbarf record) %
   (fdbarf) RECORD  % description of archive flags field %
      fdbunu[11],       % unused bits %
      fdbaar[1],        % file already archived %
      fdbadl[1],        % don't delete file after archiving %
      fdbmrk[1],        % archived but not marked complete - unused %
      fdbdmp[1],        % dumped but not marked complete - unused %
      fdbnar[1],        % don't archive this file %
      fdbarc[1];        % archive pending %
   (fdbcnr) RECORD % description of fdbcnt word in fdb %
      flnmrd[18],       % number of times this files read %
      flnmwr[18];       % number of times this file written %
   (fdbbyr) RECORD  % bits within fdbbyv word in fdb %
      flpgsz[18],       % size of file in pages %
      flfill[6],        % unused %
      flbyts[6],        % byte size %
      fldfr[6]; % default no. of versions to keep %
   (dlflbk) RECORD  % directory list file block %
      dlfbnb[18],       % pointer to next file block this group %
      dlfbpb[18],       % pointer to previous file block this group %
      dlfbgk[36],       % group key for this file %
      dlfbsk[36],       % sort key for this file %
      dlfbal[18],       % astr - link for this file %
      dlfbff[18],       % address of file fdb %
      dlfbpf[18],       % address of pc fdb %
      dlfbfl[18],       % status bits this entry (see dlfbst RECORD) %
```

```
   dlfbcp[18],          % chain pointer for second pass %
   dlfbap[18];          % astr - name of this file or its pcname %

   DECLARE EXTERNAL
      dlfbl= 6;

(dlfbst) RECORD  % definition of dlfbfl in dlflbk RECORD %
   dlstig[1],           % this entry redundant - ignore it %
   dlstpc[1],           % this file is a partial copy %
   dlstnl[1];           % this is an NLS file for which a pc exists %

(dlgrbk) RECORD  % directory list grouping block %
   dlgbnb[18],          % pointer to next block %
   dlgbpb[18],          % pointer to previous block %
   dlgbsc[18],          % pointer to start of data chain this group %
   dlgbnu[18],          % not used %
   dlgbgk[36];          % group key for this group %

   DECLARE EXTERNAL
      dlgbl= 3;

(dlmblk)  RECORD % directory list - master blocks %
   % the first 5 fields must parallel the record blkptr %
      dlmbln[11],  %length of block requested%
      dlmbal[11],  %actual length of block%
      dlmbfr[1],   %true if block is on a free list%
      dlmbpr[1],   %true if previous block is on a free list%
      dlmbfl[12],  % filler %
   dlmbzn[18],     % zone to which this master block belongs %
   dlmbnm[18],     % pointer to next master block %
   dlmbtp[18];     % free storage pointer %

   DECLARE EXTERNAL
      dlmbl= 3;

SET EXTERNAL % offsets within an fdb %
   fdbctl= 1,           % control word  (see fdbctr record) %
   fdbprt= 4,           % protection word (see fdbprr record) %
   fdbcre= 5,           % creation time & date of original vesion %
   fdbuse= 6,           % LH= last write dir. # %
   fdbver= 7,           % LH is version number %
   fdbact= 10B,         % account information %
   fdbbyv= 11B,         % see fdbbyr record %
   fdbsiz= 12B,         % byte count to end of file %
   fdbcrv= 13B,         % creation time & date this version %
   fdbwrt= 14B,         % time & date of last write %
   fdbref= 15B,         % time & date of last read %
   fdbcnt= 16B,         % see fdbcnr record %
   fdbbkf= 17B,         % see fdbbfr record %
   fdbart= 20B,         % see fdbarr record %
   fdbtdm= 21B,         % time & date of most recent dump %
   fdbtfa= 22B,         % time & date of first archive %
   fdbtsa= 23B,         % time & date of second archive %
   fdbusw= 24B,         % user settable word %
   xfdbjfn=25B;         %space for file jdn (ver. 3)%
      %used internally in NLS, see mkdirlist%
```

```
%Record Definitions%
   (card) RECORD  % substitute candidate record %
      catnc[18], %length of test string%
      carnc[18], %length of replacement string%
      catbp[36], %byte ptr to test string%
      carbp[36], %byte ptr to replacement string%
      canxt[18]; %next card for this initial char%
   (cmblock) RECORD %command block header information%
      cbprev[18],    %previous command blk addr%
      cbnxt[18],     %next command blk addr%
      cmnxt[18],     %next free entry, this block%
      cmlst[18],     %last useable entry, this blk%
      cmda[18];      %da address, this usage of commands list%

      DECLARE EXTERNAL
         cbhl= 3; %command block header length%
   (command) RECORD
      cparm1[18],
      cparm2[6],
      copcode[8],
      ctype[4];
   (corpag)  RECORD  % group allocation data page format %
      corlck[36],       %lock for this page%
      cortim[36],       %time of last locking%
      corcpy[36];       %not used%
   (corpgr) RECORD %one word.  core page status record.  gives status
   for a given core page for random files.%
      ctfull[1],     %true if the page is in use%
      ctfile[5],     %file to which the page belongs%
      ctpnum[9],     %page number within the file%
      ctfroz[3];     %number of reasons why frozen%

         %The array CORPST is the core page status table and is
         made up of instances of the above record.  RFPMAX gives
         the number of core pages that may contain file pages.
         The core pages are located at positions indicated by
         the array CRPGAD (core page address).  CORPST is indexed
         by numbers in the range [1, RFPMAX).  The starting
         location of page k is given by crpgad[k].%

   (deliverymodes) RECORD %Record containing flags for Journal delivery%
      delol[1],  %Deliver on-line%
      delnet[1], %Deliver Network%
      delhc[1];  %Deliver hard copy%
   (fhflgs) RECORD %flag word in file header%
      prvsts[1],
      fhunused[35];
   (fileblockheader) RECORD %fbhdl is length%
      fbnull[36],    %unused%
      fbind[9],     %status table index%
      fbpnum[9],    %page number in file of this block%
      fbtype[5];    %type of this block
         hdtyp  = header
         sdbtyp = data
         rngtyp = ring
         jnktyp = misc (such as keyword, viewchange etc.)%
```

```
(filstr) RECORD %File status table record.   entry length= filstl,
max no. entries = filmax%
   flexis[1],                 %true: entry represents an existant
   file%
   flhead[9],                 %crgpad index of the file header%
   flbrws[1],                 %this file in browse mode%
   fllock[1],                 %file was locked by another user when
   loaded%
   flpcread[1],       %PC read only-- write open failed (openpc)%
   flaccm[8],                 %file access mask%
   fldirno[12],       %directory number for the original file%
   finoclos[1],       % do not close this file %
   flpart[18],                %JFN for the partial copy%
   flbpart[18],       %JFN for the browse partial copy%
   florig[18],                %JFN for the original file%
   flastr[18],                %address of the file name string%
   flpcst[18],                %address of partial copy name string%
   flbpcst[18];       %address of browse partial copy name string%
   DECLARE EXTERNAL
      filstl= 4,
      filmax= 25;

(isafwa) RECORD %length is 3 words%
   isstid[36], %last completed stid%
   isbuff[18], %sequential file buffer address%
   islevs[18],  %location of string for levadj%
   isinfno[8], %sequential file number%
   istatnd[8], %character that signals end of statement%
   islevb[8], %number of leading blanks per level%
   islstb[8], %number of leading blanks, last statement%
   isdpth[8], %depth from initial psid%
   isfiltyp[1]; %TRUE if tenex file, else from 940%

(lhjfn) RECORD          % lefthalf bits from gtjfn %
   lhjrun[4],           % unused rightmost bits %
   lhjb13[1],           % bit 13: always zero %
   lhjb12[1],           % bit 12: complement of B8 in gtjfn call %
   lhjb11[1],           % bit 11: ;T given %
   lhjb10[1],           % bit 10: account given %
   lhjb9[1], % bit 9: protection given %
   lhjb8[1], % bit 8: use lowest version %
   lhjb7[1], % bit 7: use next highest version %
   lhjb6[1], % bit 6: use highest version %
   vstar[1], % asterick for version number %
   estar[1], % asterick for extension name %
   tstar[1], % asterick for file name %
   dstar[1], % asterick for directory name %
   ustar[1], % asterick for unit %
   dvstar[1],           % asterick for device %
   lfjlun[18]           % unused leftmost bits %
   ;
(lock) RECORD %lock word in file descriptor block%
   lkinit[20],          %ident of user locking the file%
   lkdirn[10],          %directory number of the file's directory%
   acctyp[3],           %access to the file%
```

```
   ojdelf[1];          %flag: new journal mail delivered%
(measrec) RECORD
   msbegin[36],  %location to begin measurement%
   msend[36],  %location to end measurements%
   msbinstn[36], %instruction displaced to begin measurement%
   mseinstn[36], %instruction displaced to end measurement%
   msmax[36], %maximum number of passes to measure%
   mscurr[36], %current number of passes measured%
   mslclk[36], %clock at beginning of interval%
   mstclk[36]; %total clock time, this series%

   DECLARE EXTERNAL
      msentl=8;

(mrker) RECORD
   mkname[36],
   mkpsid[18],
   mkfix[1],
   mkccnt[12],
   mkexis[1];
   DECLARE EXTERNAL
      mkrl= 2;

(ojtime) RECORD %on-line delivery time format%
   ojtitt[12], %hour times 64 plus fractional part of hour (64ths)%
   ojtinn[6], %number of deliveries (at ojtiii hour intervals)%
   ojtiii[6], %delivery interval, 0 defaults to 1%
   ojtijjj[9]; %slinker op flag%
(opflgs) RECORD       % format of flags word passed to OP %
   opform[1],         % TRUE then send form feeds %
   opsimff[1],        % TRUE if simulate form feeds %
   opwtpb[1];         % TRUE if wait at end of page breaks %
(rfstr) RECORD % Random file block status record. (=0 then
unallocated, else one of the following records).%

   rfexis[1],   %true if the block exists in the file%
   rfpart[1],   %true if block comes from partial copy%
   rfnull[2],   %unused%
   rfused[10],  %used word count for the block%
   rffree[10],  %free pointer for the block%
   rfcore[9];   %0 then not in core, else page index%

   % The table RFBS is broken into two sections each of which
   contains a block a records of the above type.  The first section
   includes RNGM entries from RFBS[RNGBAS] up to and including
   RFBS[RNGBAS+RNGM-1] and contains information about the ring blocks
   in the file.  The second section includes DTBM entries from
   RFBS[DTBBAS] up to and including RFBS[DTBBAS+DTBM-1] and contains
   information about the data blocks in the file.  The entry
   RFBS[RNGBAS+i] may also be referenced as RNGST[i]; likewise
   RFBS[DTBBAS+i] may be referenced as DTBST[i].  The index in RFBS
   of a block is the actual page number of the block in the file. %

   % Data blocks are allocated in the file starting with page DTBBAS.
   Up to DTBM data blocks may be allocated, with data blocks given
   internal numbers from 0 to DTBM-1.  The array DTBST in the file
```

```
   header is the data block status table and contains a one word
   record for each potential data block.  A zero entry means that the
   block does not exist in the file, otherwise the entry is as
   described in the above record defintion.  A pointer to an SDB
   (PSDB) consists of a nine bit data block number in the range
   [0,DTBM) and a nine bit displacement from the start of the block.
   The variable DTBL is maintained in each file header as the current
   upper bound on allocated data blocks for that file.  This is used
   to limit the search for a location for a new SDB.  The variable
   DTBLST contains the index of the block from which an SDB was last
   allocated or freed. %

(seqr) RECORD  % sequence generator work area -- sqwrkl words long %
   swcstid[36],   % real STID of current statement %
   swlbstid[36],  % STID of statement heading last branch in the
   sequence %
   swstid[36],    % "stid" of last item "port-sent" from this
   sequence -- it may be an ENDFIL, a stastr (pointer to an
   a-string), or the same as SWCSTID %
   swusqcod[18],  % address of user sequence generator code or 0 %
   swcacode[18],  % address of content anylyzer code or 0 %
   swvspec[36],   % first word of viewspecs %
   swvsp2[36],    % second word of viewspecs %
   swsrsav[36],   % save area for s-register when switch stacks %
   swmrsav[36],   % save area for m-register when switch stacks %
   swsvw[18],     % address of statement vector work area %
   swslvl[6],     % level at which sequence was started %
   swclvl[6],     % current statement's level %
   swklg[1],      % has first item in this sequence been port-send --
   used for k viewspec %
   swalloc[1],    % whether or not this work area is allocated %
   swcall[1],     % is user seqgen (or SSEQGEN) to be CALLed -- or
   gotten to thru the "port-send" mechanism? %
   swstkdec[1],      %True if there is a stack declared for this work
   area%
   swstkalloc[1],    %whether or not the stack has been allocated %
   swfill[1],              %currently unused %
   swstkloc[18],     % address of the stack associated with this
   sequence work area or 0 implying a stack must be
   allocated%
   swstksz[18];            %size of stack associated with this
   sequence%
   SET EXTERNAL
      sqwrkl= 10;

(shed) RECORD  % substitute header record %
   sbrp[36], %pointer to next space for card%
   sbdp[36], %pointer to dispatcher%
   sbas[36], %pointer to A-string for strings%
   sbtt[36]; %type of entity being substituted%
      %this field can be shortened; only needs to hold numbers up to
      32, at the outside%
(substr) RECORD %substitute stack entry record%
   sbc1cnt[12],
   sbc2cnt[12],
   sbc3cnt[12];
```

```
(tenexbits)RECORD %Bits according to the TENEX way%
   bitfiller[32],
   bit3[1],
   bit2[1],
   bit1[1],
   b0[1];
(usrblk)  RECORD  %group allocation userjob entry record%
   usrlnk[36],       %lh= back link to previous user block%
             %rh= forward link%
   usrno[18],        %user directory number(rh!)%
   usrflgs[18],      %flags (see defs)(lh!)%
   usrgrp[36],       %user group number%
   usrjob[36],       %user job number%
   usrtty[36],       %user tty number(-1 if detached)%
   usrtod[36],       %time logged in(for connectime)%
   usrrt[36],        %runtime for this job%
   usrtim[36];       %todclk of last runtime update%
DECLARE EXTERNAL
   lcard= 4; %length of card in words%

DECLARE EXTERNAL
   lshed= 4; %length of shed in words%

FINISH
```

CALC SUPPORT

```
< NLS, CALCSUPPORT.NLS;3, >, 3-AUG-76 23:24 KJM ;;;;
FILE calcsupport % L10 <REL-NLS>CALCSUPPORT.rel %% (110,)
(REL-NLS,CALCSUPPORT.rel,) %
% DECLARATIONS %
   REGISTER
      r0 = 0, r1 = 1, r2 = 2, r3 = 3, r4 = 4, r5 = 5, r6 = 6, a1 = 12,
      a2 = 13, a3 = 14, a4 = 15;

   DECLARE FIELD
      expf=[3B,9:27];

   DECLARE EXTERNAL
      fone=2014B8;
   DECLARE
      ften=2045B8, ften1=151B9,
      ftenth=1756314631468, ften11=142314631463B;

% ARITHMETIC %
   (qcmult) PROCEDURE (ar,ma);   % floating multiply ar _ ar * ma %
      %parameters are all addresses%
      !MOVE r3,ma;
      !MOVE r2,ar;
      !MOVE r4,0(r2);
      !MOVE r5,1(r2);
      !MOVEM r4,r6;
      !FMPR r6,1(r3);
      !FMPR r5,0(r3);
      !UFA r5,r6;
      !FMPL r4,0(r3);
      !UFA r5,r6;
      !FADL r4,r6;
      !MOVEM r4,0(r2);
      !MOVEM r5,1(r2);
      RETURN; END.

   (qcadd) PROCEDURE (ar,ma);   % floating add ar _ ar + ma %
      !MOVE r3,ma;
      !MOVE r2,ar;
      !MOVE r4,0(r2);
      !MOVE r5,1(r2);
      !UFA r5,1(r3);
      !FADL r4,0(r3);
      !UFA r5,r6;
      !FADL r4,r6;
      !MOVEM r4,0(r2);
      !MOVEM r5,1(r2);
      RETURN END.

   (qcdiv) PROCEDURE (ar,ma);   % floating divide ar _ ar / ma %
      LOCAL quo[2];
      REF ar, ma;
      qcdivw(&ar,&ma,$quo);
      ar _ quo;
      ar[1] _ quo[1];
      RETURN; END.
```

```
(qcdivw) PROCEDURE (ar,ma,quo);   % floating divide quo _ ar / ma %
   %parameters are all addresses%
   !MOVE r3,ma;
   !MOVE r2,ar;
   !MOVE r4,0(r2);
   !MOVE r5,1(r2);
   !FDVL r4,0(r3);
   !MOVN r6,r4;
   !FMPR r6,1(r3);
   !UFA r5,r6;
   !FDVR r6,0(r3);
   !FADL r4,r6;
   !MOVE r2,quo;
   !MOVEM r4,0(r2);
   !MOVEM r5,1(r2);
   RETURN END.

(qcsub) PROCEDURE (ar,ma);   % floating subtract ar _ ar - ma %
   !MOVE r3,ma;
   !MOVE r2,ar;
   !MOVE r4,0(r2);
   !MOVE r5,1(r2);
   !DFN r4,r5;
   !UFA r5,1(r3);
   !FADL r4,0(r3);
   !UFA r5,r6;
   !FADL r4,r6;
   !DFN r4,r5;
   !MOVEM r4,0(r2);
   !MOVEM r5,1(r2);
   RETURN; END.

(qcneg) PROCEDURE (ar);   % floating negate ar _ -ar %
   !MOVE r2,ar;
   !MOVE r4,0(r2);
   !DFN r4,1(r2);
   !MOVEM r4,0(r2);
   RETURN; END.

(qfloat) PROCEDURE (ar,ch);   % convert character (ascii 0-9) to
floating point number %
   %ar is address of result%
   LOCAL expa;
   REF ar;

   expa _ 200B;
   r3_0;
   .expf_(ch-"0");
   WHILE SKIP !TLNN r3,777000B DO
      BEGIN
      !LSH r3,-1;
      BUMP expa;
      END;
   .expf_expa;
   ar_r3;
   BUMP &ar;
```

```
    r3_0;
    .expf_expa-27;
    ar_r3;
    RETURN; END.

(nfloat) PROCEDURE (instring,fl1,fl2);   % convert string to floating
point number %
    %convert character string to double precision floating point%
    %instring is address of input string, fl1, fl2 will contain
    results%
    LOCAL chr,fchr,fchr1,term,term1,sflg;
    REF fl1, fl2, instring;

    %Set up flag for negative umber%
    sflg _ FALSE;
    %set READC pointer to head of string%
    CCPOS  SF(*instring*);
    fl1 _ fl2 _ 0; %clear result area%
    %take care of portion to left of decimal point%
    CASE (chr _ READC) OF
        IN ['0,'9]:
            BEGIN
            qcmult(&fl1, $ften);
            qfloat($fchr,chr);
            qcadd(&fl1, $fchr);
            REPEAT CASE;
            END;
        = '-:
            BEGIN
            sflg _ TRUE;
            REPEAT CASE;
            END;
        = '.:
            %take care of portion to right of decimal point%
            BEGIN
            term_fone;
            term1_0; %temporaries%
            CASE (chr _ READC) OF
                IN ['0,'9]:
                    BEGIN
                    qcmult($term,$ftenth);
                    qfloat($fchr,chr);
                    qcmult($fchr,$term);
                    qcadd(&fl1,$fchr);
                    REPEAT CASE;
                    END;

                = '-:
                    BEGIN
                    sflg _ TRUE;
                    REPEAT CASE;
                    END;

                = ENDCHR:
                    EXIT CASE;
```

```
            ENDCASE REPEAT CASE; %drop editing characters%
            END;

        = ENDCHR: EXIT CASE;

        ENDCASE REPEAT CASE; %drop editing characters%

    RETURN (sflg);
    END.

FINISH
```

CATNUM

```
< NLS, CATNUM.NLS;5, >, 7-JAN-77 17:17 JDH ;;;;( NLS, CATNUM.NLS;36, ),
23-MAY-74 12:58 HGL ;

FILE catnum   % L10 to <REL-NLS>catnum %% (L10,) (rel-nls,catnum.rel,) %
                                                                      02
%General Number System Library Routines%                              0101
   (gcatnums) %Get count catalog numbers for subcollection(s) substr.
   type is passed to getcnum. Return numbers (separated by SP's) in
   retstr if Non-zero%
   PROCEDURE (retstr, idntsr,  substr, count, type);                  0420
      LOCAL numfstid;                                                 0421
      LOCAL STRING cnumber[10];                                       0422
      REF retstr, substr, idntsr;                                     0423
                                                                      0424
      IF &retstr THEN *retstr* _ NULL;                                0425
      conjdir(TRUE);                                                  0426
      numfstid _ 0;                                                   0427
      ON SIGNAL ELSE                                                  0428
         BEGIN                                                        0429
         IF numfstid.stfile THEN sigclose(numfstid.stfile := 0);      0430
         ON SIGNAL ELSE;                                              0431
         conjdir(FALSE);                                              0432
         END;                                                         0433
      numfstid _ openlock(0, jflname($"tcnumbers"));                  0434
      WHILE (count _ count-1) >= 0 DO                                 0435
         BEGIN                                                        0436
         %now assign a number, and mark it used%                      0437
            getcnum(&idntsr, $cnumber, &substr, type, numfstid);      0438
            IF &retstr THEN *retstr* _ *retstr*, *cnumber*, SP;       0439
         END;                                                         0440
      close(numfstid.stfile := 0);                                    0441
      ON SIGNAL ELSE RETURN (TRUE, TRUE);                             0442
         %second return indicates reconnect error, message addr in
         sysmsg%                                                      0443
      conjdir(FALSE);                                                 0444
      RETURN(TRUE, FALSE);                                            0445
      END.                                                            0446
                                                                      0447
   (ckcnum) %This procedure accepts a catalog number in *number*, an
   identification-list in *idlist*, and searches the preassigne-number
   list for that number.%
   PROCEDURE (type, number, idlist, nstid);                           0127
      % The action it takes depends on the parameter TYPE:           0128
         =0: if the number is preassigned to this user then           0129
            return true                                               0130
         =1: if the number is preassigned to this user and not inuse
         then                                                         0131
            mark number as in use                                     0132
            return true                                               0133
         =2: if the number is preassigned to this user and inuse then
                                                                      0134
            mark not in use                                           0135
            return true                                               0136
      Assumes connected to journal directory.                         0137
      %                                                               0138
      LOCAL stid, savestid, retval, numstid;                          0139
```

```
   LOCAL TEXT POINTER  z1, z2;                                          0140
   LOCAL STRING reservelist[50], idstr[20];                             0141
   REF number, idlist;                                                  0142
                                                                        0143
   numstid _ 0;                                                         0144
   %Set up SIGNAL for errors%                                           0145
      ON SIGNAL ELSE                                                    0146
         IF NOT nstid AND numstid.stfile THEN                           0147
            sigclose(numstid.stfile := 0);                              0148
   %open number file, and look for number%                              0149
      numstid _ IF nstid THEN nstid                                     0150
         ELSE  openlock(0, jflname($"tcnumbers"));                      0151
      enablaccess(numstid.stfile, jrnlaccess);                          0152
      IF NOT numtype(numstid, &number, $"PREASSIGNED": savestid) THEN
                                                                        0153
         err($"Number Not Reserved");                                   0154
      IF NOT FIND SF(savestid) [']] $NP $PT $NP ^z1 ([';] /  [D '-] <
      $PT > / [NP]) ^z2 _z2 THEN                                        0155
         BEGIN                                                          0156
         lockjo(1);                                                     0157
         err($"Illegal number file format -- Please report this to
         the NIC");                                                     0158
         END;                                                           0159
      *reservelist* _ +z1 z2; %ident field%                             0160
      FIND SF(*idlist*) ^z2 ^z1;                                        0161
      LOOP %process list of idents%                                     0162
         BEGIN                                                          0163
         FIND z2 > $(SP/',) ('( [']] $(SP/',) /) $('&/'^) ^z1
         $(LD/'-) ^z2;                                                  0164
         *idstr* _ +z1 z2;                                              0165
         IF NOT idstr.L THEN EXIT LOOP; %end of ident list%             0166
         IF FIND SF(*reservelist*) [*idstr*] THEN                       0167
            BEGIN                                                       0168
            CASE type OF                                                0169
               = 0: %number is reserved by this user%                   0170
                  retval _ TRUE;                                        0171
               = 1: %mark as in-use%                                    0172
                  IF retval _ ( NOT FIND SF(savestid) [" INUSE "] )
                  THEN                                                  0173
                     ST savestid _ SF(savestid) SE(savestid), " INUSE
                     "                                                  0174
               = 2:  %mark as not-in-use%                               0175
                  IF retval _ (FIND SF(savestid) [" INUSE "] ^z2 <
                  ['I] CH ^z1) THEN                                     0176
                     ST z1 z2 _ NULL;                                   0177
               ENDCASE err ($"Illegal operation type encountered in
               ckcnum");                                                0178
            IF NOT nstid THEN close(numstid.stfile := 0);               0179
            RETURN(retval, savestid);                                   0180
            END;                                                        0181
         END;                                                           0182
      err($"number reserved by someone else!");                         0183
   END.
                                                                        0184
(ckrfcnum) %This procedure accepts a catalog number in *number*, an
RFC number in *rfcnumber*, an identification-list in *idlist*, and
```

```
searches the preassigne-number list for that number.%
PROCEDURE (type, rfcnumber, number, idlist, rstid, nstid);          0185
   % The action it takes depends on the parameter TYPE:            0186
      =0: if the numbers are preassigned to this user then         0187
         return true                                               0188
      =1: if the numbers are preassigned to this user and not inuse
      then                                                         0189
         mark numbers as in use                                    0190
         return true                                               0191
      =2: if the numbers are preassigned to this user and inuse then
                                                                   0192
         mark not in use                                           0193
         return true                                               0194
   Assumes connected to journal directory.                         0195
   %                                                               0196
   LOCAL stid, nsavstid, rsavstid, retval, numstid, rnumstid;      0197
   LOCAL TEXT POINTER  z1, z2;                                     0198
   LOCAL STRING rfcreservelist[50], reservelist[50], idstr[20];    0199
   REF number, rfcnumber, idlist;                                  0200
                                                                   0201
   numstid _ rnumstid _ 0;                                         0202
   %Set up SIGNAL for errors%                                      0203
      ON SIGNAL ELSE                                               0204
         BEGIN                                                     0205
         IF NOT nstid AND numstid.stfile THEN                      0206
            sigclose(numstid.stfile := 0);                         0207
         IF NOT rstid AND rnumstid.stfile THEN                     0208
            sigclose(rnumstid.stfile := 0);                        0209
         END;                                                      0210
   %open number file, and look for number%                         0211
      numstid _ IF nstid THEN nstid                                0212
         ELSE  openlock(0, jflname($"tcnumbers"));                 0213
      rnumstid _ IF rstid THEN rstid                               0214
         ELSE  openlock(0, jflname($"RFCNUMBERS"));                0215
      enablaccess(numstid.stfile, jrnlaccess);                     0216
      enablaccess(rnumstid.stfile, jrnlaccess);                    0217
      IF NOT numtype(rnumstid, &rfcnumber, $"PREASSIGNED": rsavstid)
      THEN                                                         0218
         err($"RFC Number Not Reserved");                          0219
      IF NOT numtype(numstid, &number, $"PREASSIGNED": nsavstid) THEN
                                                                   0220
         err($"Catalog Number Not Reserved");                      0221
      IF NOT FIND SF(rsavstid) [")] $NP $PT $NP ^z1 ([";] /  [D "-] <
      $PT > / [NP]) ^z2 _z2 THEN                                   0222
         BEGIN                                                     0223
         lockjo(1);                                                0224
         err($"Illegal RFC number file format -- Please report this
         to the NIC");                                             0225
         END;                                                      0226
      *rfcreservelist* _ +z1 z2; %ident field%                     0227
      IF NOT FIND SF(nsavstid) [")] $NP $PT $NP ^z1 ([";] /  [D "-] <
      $PT > / [NP]) ^z2 _z2 THEN                                   0228
         BEGIN                                                     0229
         lockjo(1);                                                0230
         err($"Illegal Catalog number file format -- Please report
         this to the NIC");                                        0231
```

```
      END;                                                          0232
   *reservelist* _ +z1 z2; %ident field%                            0233
   FIND SF(*idlist*) ^z2 ^z1;                                       0234
   LOOP %process list of idents%                                    0235
      BEGIN                                                         0236
      FIND z2 > $(SP/',) ('( [')] $(SP/',) /) ^z1 $(LD/'-) ^z2;
                                                                    0237
      *idstr* _ +z1 z2;                                             0238
      IF NOT idstr.L THEN EXIT LOOP; %end of ident list%            0239
      IF (FIND SF(*rfcreservelist*) [*idstr*]) AND (FIND
      SF(*reservelist*) [*idstr*]) THEN                             0240
         BEGIN                                                      0241
         CASE type OF                                               0242
            = 0: %number is reserved by this user%                  0243
               retval _ TRUE;                                       0244
            = 1: %mark as in-use%                                   0245
               BEGIN                                                0246
               IF retval _ ( NOT FIND SF(rsavstid) [" INUSE "] )
               THEN                                                 0247
                  ST rsavstid _ SF(rsavstid) SE(rsavstid), " INUSE
                  ";                                                0248
               IF NOT FIND SF(nsavstid) [" INUSE "] THEN            0249
                  ST nsavstid _ SF(nsavstid) SE(nsavstid), " INUSE
                  "                                                 0250
               END;                                                 0251
            = 2:  %mark as not-in-use%                              0252
               BEGIN                                                0253
               IF retval _ (FIND SF(rsavstid) [" INUSE "] ^z2 <
               ['I] CH ^z1) THEN                                    0254
                  ST z1 z2 _ NULL;                                  0255
               IF FIND SF(nsavstid) [" INUSE "] ^z2 < ['I] CH ^z1
               THEN                                                 0256
                  ST z1 z2 _ NULL;                                  0257
               END;                                                 0258
            ENDCASE err ($"Illegal operation type encountered in
            ckcnum");                                               0259
         IF NOT nstid THEN close(numstid.stfile := 0);              0260
         IF NOT rstid THEN close(rnumstid.stfile := 0);             0261
         RETURN(retval, rsavstid, nsavstid);                        0262
         END;                                                       0263
      END;                                                          0264
   err($"Number reserved by someone else!");                        0265
END.
                                                                    0266
(getcnum) %This is the routine which is used for reserving catalogue
numbers%
PROCEDURE (identsr, numbsr, typesr, destination, stid);             0267
   % Formal Parameters:                                            0268
      IDENTSR is a string containing the ident of the requestor of
      the number                                                    0269
      NUMBSR is the string which will be used to return the reserved
      number                                                        0270
      TYPESR contains the address of a string which identifies the
      process requesting the number (e.g. JOURNAL or XDOC)          0271
      DESTINATION contains: 0 if number is to be moved to 'IN USE
      branch, 1 for 'PREASSIGNED', and 2 for 'USED'%                0272
```

```
   LOCAL  count, numstid, trap, capsav;                                0273
   LOCAL TEXT POINTER z1, z2, z3, z4;                                  0274
   LOCAL STRING tempsr[15];                                            0275
   REF identsr, numbsr, typesr;                                        0276
                                                                       0277
   %assumes connected to journal directory%                            0278
   %Abort if journal not in operation%                                 0279
      IF jolock() THEN                                                 0280
         werr($"number system temporarily unavailable");               0281
   %Set up SIGNAL for errors%                                          0282
      numstid _ 0;                                                     0283
      trap _ FALSE;                                                    0284
      ON SIGNAL ELSE                                                   0285
         BEGIN                                                         0286
         IF trap THEN notrapcc(capsav);                                0287
         IF stid = 0 THEN sigclose(numstid.stfile := 0);               0288
         END;                                                          0289
   IF stid THEN                                                        0290
      BEGIN                                                            0291
      numstid _ stid;                                                  0292
      numstid.stpsid _ origin;                                         0293
      END                                                              0294
   ELSE numstid _ openlock(0, jflname($"tcnumbers"));                  0295
   enablaccess(numstid.stfile, jrnlaccess); %let him write it%         0296
   IF (numstid _ namelook(numstid, $"FREE")) = endfil THEN             0297
      err($"Illegal Number File Format -- report to NIC");             0298
   IF NOT FIND SF(numstid) [^C/D] ^z1 THEN                             0299
      err($"Number File Exhausted -- report to NIC");                  0300
   %Check to see if it is an interval%                                 0301
   trap _ TRUE;                                                        0302
   capsav _ trapcc();                                                  0303
   IF FIND z1 < ^C > THEN                                              0304
      BEGIN                                                            0305
      %it is an interval, so we will have to work a little%            0306
      FIND z1 $D ^z2 %First number%                                    0307
         $(-D) ^z3 $D ^z4; %second number%                             0308
         *numbsr* _ z1 z2;                                             0309
         *tempsr* _ z3 z4;                                             0310
         IF *tempsr* = *numbsr*  THEN %equal...delete whole schmere%
                                                                       0311
            BEGIN                                                      0312
            FIND z1 _z1 [^J] ^z2;                                      0313
            ST z1 z2 _ NULL;                                           0314
            END                                                        0315
         ELSE                                                          0316
            BEGIN %add one from first number %                         0317
            *tempsr* _ *numbsr*;                                       0318
            bumpstr($tempsr);                                          0319
            ST z1 z2 _ *tempsr*;                                       0320
            END;                                                       0321
      END                                                              0322
   ELSE                                                                0323
      BEGIN %single number%                                            0324
      FIND > _z1 z1 $D ^z2;                                            0325
      *numbsr* _ z1 z2;                                                0326
      ST z1 z2 _ NULL;                                                 0327
```

```
      END;                                                          0328
   %Now move number entry to proper list%                           0329
      CASE destination OF                                           0330
         = 2: *tempsr* _ "USED";                                    0331
         = 1: *tempsr* _ "PREASSIGNED";                             0332
         ENDCASE *tempsr* _ "INUSE";                                0333
      %move it to the branch%                                       0334
         numstid _ movecnum($tempsr, &numbsr, &identsr, &typesr,
         numstid);                                                  0335
   notrapcc(capsav);                                                0336
   trap _ FALSE;                                                    0337
   IF NOT stid THEN close(numstid.stfile := 0);                     0338
   RETURN(numstid);                                                 0339
   END.
                                                                    0340
(movecnum) %Move designated number under deisgnated list (branch
name). If the list doesn't exist, create it%
PROCEDURE (brname, numbsr, identsr, typesr, stid);                  0341
   LOCAL tstid;                                                     0411
   LOCAL TEXT POINTER z1, z2;                                       0342
   LOCAL STRING tempsr[200];                                        0343
   REF brname, numbsr, identsr, typesr;                             0344
                                                                    0345
   IF (tstid _ namelook(stid, &brname)) = endfil THEN               0346
      BEGIN %branch doesn't exist..create it%                       0347
      *tempsr* _ '(, *brname*, ") ";                                0348
      FIND SF(*tempsr*) ^z1 SE(*tempsr*) ^z2;                       0349
      stid _ cinssta(getail(getsub(stid)), levsuc, $z1, $z2);       0350
      END                                                           0351
   ELSE stid _ tstid;                                               0352
   *tempsr* _ NULL;                                                 0353
   dtfrmt(-1, $tempsr);                                             0355
   astruc(&identsr);                                                0356
   *tempsr* _ "(C",*numbsr*,  ") ", *typesr*, SP,  *identsr*, "; ",
   *tempsr*;                                                        0357
   FIND SF(*tempsr*) ^z1 SE(*tempsr*) ^z2;                          0358
   stid _ cinssta(stid, levdown, $z1, $z2);                         0359
   RETURN(stid);                                                    0360
   END.                                                             0361
                                                                    0362
(numtype) %Compares name of branch containing numstr to string
typesr, and returns true if same, false if not.  Returns stid of
number statement as second result%
PROCEDURE (stid, numstr, typesr);                                   0363
   LOCAL savestid, retstid;                                         0364
   LOCAL STRING tempsr[50];                                         0365
   REF numstr, typesr;                                              0366
                                                                    0367
   *tempsr* _ 'C, *numstr*;                                         0368
   IF (savestid _ retstid _ namelook(stid, $tempsr)) = endfil THEN
                                                                    0369
      err($"Number not reserved!");                                 0370
   %Now check that it is in right branch%                           0371
      WHILE (stid _ getup(savestid)).stpsid # origin DO savestid _
      stid;                                                         0372
      CCPOS SF(savestid);                                           0373
```

```
      xtrnam($tempsr, $swork, '(, ')); %read name into tempsr%   0374
   RETURN(*tempsr* = *typesr*, retstid);                         0375
   END.
                                                                 0376
(usedcnum) %This routine moves the number in numbsr from the in-use
or pre-assigned list to the used list%
PROCEDURE (numbsr, nstid);                                       0377
   LOCAL numstid, stid, xstid;                                   0378
   LOCAL TEXT POINTER z1;                                        0379
   LOCAL STRING tempsr[100];                                     0380
   REF numbsr;                                                   0381
                                                                 0382
   %Set up SIGNAL for errors%                                    0383
      numstid _ 0;                                               0384
      ON SIGNAL ELSE                                             0385
         IF nstid = 0 THEN sigclose(numstid.stfile := 0);        0386
   numstid _ IF nstid THEN nstid                                 0387
      ELSE openlock(0, jflname($"tcnumbers"));                   0388
   enablaccess(numstid.stfile, jrnlaccess);                      0389
   %now find number%                                             0390
      *tempsr* _ 'C, *numbsr*;                                   0391
      IF (stid _ namelook(numstid, $tempsr)) = endfil THEN       0392
         BEGIN                                                   0393
         *tempsr* _ *numbsr*, " Not Reserved";                   0394
         err($tempsr);                                           0395
         END;                                                    0396
      numstid _ stid; %stid of statement%                        0397
      WHILE (xstid _ getup(stid)).stpsid # origin DO stid _ xstid;
                                                                 0398
      %by here, stid contains head of branch...check name%      0399
      CCPOS SF(stid);                                            0400
      xtrnam($tempsr, $swork, '(, ')); %read statement name into
      tempsr%                                                    0401
      IF (*tempsr* # "INUSE") AND (*tempsr* # "PREASSIGNED") THEN
                                                                 0402
         err($"Number neither reserved nor in use -- command
         aborted.");                                             0403
      IF (stid _ namelook(stid, $"USED")) = endfil THEN err($"Bad
      USED branch missing from the Number File -- report to NIC");
                                                                 0404
      stid _ cmovsta(stid, levdown, numstid, FALSE, 0); %move
      statement%                                                 0405
      %Now strip off in use if necessary%                        0406
         IF FIND SF(stid) ["INUSE"] < ['I] ^z1 > THEN ST stid _
         SF(stid) z1;                                            0407
   IF NOT nstid THEN close(numstid.stfile := 0);                 0408
   RETURN END.
                                                                 0409
%Support for RFC numbers%                                          03
   (rfcex) %reserve an RFC (Request For Comments) and a Journal number
   given the currently available info contained in the string pointed to
   by STID.  Reserve it on behave of the user whose ident is in authrsr%
   PROCEDURE (authorstr, titlestr, sendstr, onlineflag, rfcnum, catnum);
                                                                   04
   LOCAL STRING tempsr[250];                                       05
   LOCAL rnumstid;                                                 06
```

```
   REF authorstr, titlestr, sendstr, rfcnum, catnum;                    07
   %-----------------%                                                  08
   rnumstid _0;                                                         09
   ON SIGNAL ELSE                                                      010
      BEGIN                                                            011
      IF rnumstid.stfile THEN sigclose(rnumstid.stfile := 0);          012
      conjdir(FALSE);                                                  013
      END;                                                             014
   conjdir(TRUE);                                                      015
   %Get RFC Number%                                                    016
      %open RFC Number file%                                           017
         rnumstid _ openlock(0, jflname($"RFCNUMBERS"));               018
      rnumstid _ getcnum(&authorstr, $rfcnum, $" NWG/RFC ", 1,
      rnumstid);                                                       019
   %Get catalog number%                                                020
      getcnum(&authorstr, $catnum, $" NWG/RFC ", 1, 0);                021
   %Now fix up rfc number file to reflect all of the desired info%
                                                                       022
      IF onlineflag THEN                                              0412
         *tempsr* _                                                   0413
            "Online
            Distribution: ", *sendstr*, EOL, "Title: ", *titlestr*
                                                                       0414
      ELSE                                                            0415
         *tempsr* _                                                   0416
            "Offline
            Distribution: ", *sendstr*, EOL, "Title: ", *titlestr*;
                                                                       0417
      ST SF(rnumstid) _ SF(rnumstid) SE(rnumstid), EOL,                 023
         '#, *catnum*, EOL, *tempsr*;                                  024
      close(rnumstid.stfile := 0);                                     026
      conjdir(FALSE);                                                  027
   %Type onto logging tty%                                             028
      *tempsr* _ *catnum*, " RFC#", *rfcnum*;                          029
      jlog($tempsr, $"Pre-Assigned RFC", &authorstr);                  030
   RETURN;                                                             035
   END.
                                                                       036
                                                                       037
(rfcsyscheck) %see if the RFC system is enabled for this host%
PROCEDURE;                                                             038
   IF lhostn # rfchost THEN                                            039
      err($"RFC Numbers Not Available on This Host");                  040
   RETURN;                                                             041
   END.                                                                042
FINISH                                                                0410
```

CEDIT 1

```
< NLS, CEDIT1.NLS.47, >, 23-Apr-78 19:52 JDH ;;;;
FILE cedit1 %  L10  <REL-NLS>cedit1  % % (110,) (rel-nls,cedit1.rel,) %
                                                                      02
   DECLARE EXTERNAL                                                   03
      notyet = 7, newversion = 1, oldversion = 2, upcompact = 3, newname
      = 4;                                                            04
   REGISTER r1=1, r2=2, r3=3, a1=12, a2=13;                           05
   REF tda;                                                           06
%append%                                                             035
   (cappsta)   % GB: core NLS Append Statement procedure %
   PROCEDURE (tostid, fromstid, tp1, tp2 % => no value %);          08144
      % Procedure description                                       08145
         FUNCTION                                                   08236
            Appends the statement at 'fromstid' to the end of the
            statement at 'tostid'.  A string may be specified to insert
            between the two statements when they are appended.  (This is
            symmetric with 'cbresta'.)  The statement at 'fromstid' is
            deleted.                                                08237
         ARGUMENTS                                                  08238
            tostid - STID - statement to form beginning of new statement
                                                                    08255
            fromstid - STID - statement to form end of new statement
                                                                    08256
            tp1 - TEXT POINTER - start of string to insert          08244
            tp2 - TEXT POINTER - end of string to insert            08245
         RESULTS                                                    08246
            none                                                    08247
         NON-STANDARD CONTROL                                       08248
            Error if 'fromstid' and 'tostid' are the same statement.
                                                                    08249
         GLOBALS                                                    08250
            levdown - read only                                     08251
         EXAMPLE                                                    08252
            cappsta(stid1, stid2, $tp1, $tp2)                       08253
         %                                                          08254
      % Declarations %                                             08159
         LOCAL srsub, tosub, level;                                  038
         REF tp1, tp2;                                               039
        %msnst();%                                                08163
      IF tostid = fromstid THEN err($"illegal append");           08164
      IF (srsub _ getsub(fromstid)) # fromstid THEN               08165
         BEGIN                                                    09712
         movtst( tostid, srsub, getail(srsub));                   08166
         IF (tosub _ getsub(tostid)) # tostid THEN                09714
            BEGIN                                                 09715
            tosub _ getail(tosub);                                09716
            level _ levsuc;                                       09717
            END                                                   09718
         ELSE level _ levdown;                                    09719
         cmovgro( tosub, level, srsub, getail(srsub), FALSE, 0);  08168
         END;                                                     09713
      ST tostid _ SF(tostid) SE(tostid), tp1 tp2, SF(fromstid)
      SE(fromstid);                                               08169
      cdelgro(fromstid, fromstid, FALSE, 0);                      08170
        %msfst();%                                                08171
      RETURN;                                                     08172
```

```
END.  %%
```

```
                                                                    08162
(capptex)   % GB: core NLS Append Text procedure %
PROCEDURE (stid, t1, t2, l1, l2 % => no value %);                   08654
   % Procedure description                                          08655
      FUNCTION                                                      08656
         Appends the text at 'l1' through 'l2' onto the end of the
         statement at 'stid'.  Then appends the text at 't1' through
         't2' onto the end of that.  (This is similar to 'cappsta'.)
                                                                    08657
      ARGUMENTS                                                     08658
         stid - STID - statement to form beginning of new statement
                                                                    09370
         t1 - TEXT POINTER - start of string to insert second       09372
         t2 - TEXT POINTER - end of string to insert second         09373
         l1 - TEXT POINTER - start of string to insert first        09374
         l2 - TEXT POINTER - end of string to insert first          09375
      RESULTS                                                       08660
         none                                                       08661
      NON-STANDARD CONTROL                                          08662
         none                                                       08663
      GLOBALS                                                       08664
         none                                                       08665
      EXAMPLE                                                       08666
         capptex(stid, $tp3, $tp4, $tp1, $tp2)                      08667
      %                                                             08668
   % Declarations %                                                08669
      REF t1, t2, l1, l2;                                            053
   %msnst();%                                                       08673
   ST stid _ SF(stid) SE(stid), l1 l2, t1 t2;                       08674
   %msfst();%                                                       08675
   RETURN;                                                          08676
   END.   %%
```

```
                                                                    08672
%archive%                                                            059
   (carctil)   % UB: core NLS Archive File procedure %
   PROCEDURE (rhstn, fname, arcparms, astr % => no value %);        08677
      % Procedure description                                      08678
         FUNCTION                                                 08679
            none                                                  08680
         ARGUMENTS                                                08681
            rhstn - INTEGER -                                     08700
               number of the host on which the files reside.  Only the
               local host ('lhostn') is currently implemented.    08701
            fname - STRING -                                      09707
               file group name string in TENEX format.  Connected
               directory is assumed if a directory is not specified.
               Examples:                                          09708
                  "*.*"                                           09709
                  "<SMITH>*.PC"                                   09710
                  "<SMITH>FOO.NLS;3"                              09711
            arcparms - INTEGER - request parameters               08698
            astr - STRING - string to receive list of archived files
                                                                  09706
         RESULTS                                                  08683
            none                                                  08684
         NON-STANDARD CONTROL                                     08685
            none                                                  08686
         GLOBALS                                                  08687
            none                                                  08688
         EXAMPLE                                                  08689
            none                                                  08690
         %                                                        08691
      % Declarations %                                            08692
         LOCAL                                                    06804
            fjfn,        % main jfn %                             06805
            jfnflgs,     % left half flags for a group jfn %      06806
            pcjfn,       % jfn of partial copy %                  06807
            flags;       % bits indicating * typed for file name fields %
                                                                  06808
         LOCAL STRING                                             06810
            uname[200],          % complete name as entered by user %
                                                                  06811
            udirname[40],        % user input directory name %    06812
            ufilname[40],        % user input file name %         06813
            uextname[40],        % user input extension name %    06814
            uverno[40],          % user input version number %    06815
            ulftovr[40],         % user input beyond version number %
                                                                  06816
            jfnname[200],        % complete name from jfns %      06817
            errstring[200],      % eror message string %          06818
            tstring[200];        % temporary string %             06819
         REF fname, astr;                                         03885
      % initial variables %                                       06822
         fjfn _ jfnflgs _ pcjfn _ flags _ 0;                      06823
      % find out if remote or local (disk) file %                 06824
         CASE rhstn OF                                            06825
            = lhostn:  NULL;                                      06826
            ENDCASE err( $"remote file manipulations not implemented
```

```
      yet" );                                                    06827
   % now parse user input strings %                              06828
     parseinput( &fname, $udirname, $ufilname, $uextname, $uverno,
     $ulttovr, $uname, $flags);                                 06829
   % get main jfn (old file only, group jfn) %                   06830
     IF NOT (fjfn _ sgtjfn( gtjoif .V gtjstr + 0, $uname, $errstring
     : jfnflgs ) )THEN err( $errstring );                       06831
   % now find out if we support this type of file %              06832
     IF NOT chkdev( fjfn ) THEN err( $"only disk files supported" );
                                                                06833
   % now loop to get all files in the group %                    06834
     LOOP                                                       06835
        BEGIN                                                   06836
        % check to make sure this not a PC with * for extension name
        %                                                       06837
           IF NOT chkpcs( fjfn, flags) THEN GOTO parcnxjfn;     06838
        % get actual file name into string for use later (maybe) %
                                                                06839
           r3 _ 011110B6    %  directory file ext ver %         06840
              + (IF jfnflgs.lhjb9  THEN 1B6 ELSE 0) % ;Protection %
                                                                06841
              + (IF jfnflgs.lhjb10 THEN 1B5 ELSE 0) % ;Account %
                                                                06842
              + (IF jfnflgs.lhjb11 THEN 4B4 ELSE 0) % ;T %      06843
              + 1;          % with proper file punctuation %   06844
           jfnflink( fjfn, $jfnname, r3);                       06845
        % get PC jfn and find out if this user can access this PC %
                                                                06846
           IF NOT getpcjfn( fjfn, $pcjfn, $errstring, FALSE) THEN
                                                                06847
              BEGIN % this user can't access this file %        06848
              *tstring* _ *jfnname*, " archive status not changed
              because ", *errstring*;                           06849
              IF flags THEN                                     06850
                 BEGIN                                          06851
                 *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
                                                                06852
                 GOTO parcnxjfn;                                06853
                 END                                            06854
              ELSE                                              06855
                 BEGIN                                          06856
                 IF NOT SKIP !rljfn( fjfn) THEN NULL;           06857
                 err( $tstring );                               06858
                 END;                                           06859
              END;                                              06860
        % now set archive status for file (and its partial copy) %
                                                                06861
           IF NOT arcflandpc( fjfn, pcjfn, arcparms, $errstring)
           THEN                                                 06862
              BEGIN % can't set archive status for file (or pc) %
                                                                06863
              *tstring* _ *jfnname*, " archive status not changed
              because ", *errstring*;                           06864
              IF flags THEN                                     06865
                 BEGIN                                          06866
                 *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
```

```
                                                                    06867
                GOTO parcnxjfn;                                     06868
                END                                                 06869
             ELSE                                                   06870
                BEGIN                                               06871
                IF NOT SKIP !rljfn( fjfn) THEN NULL;                06872
                IF NOT SKIP !rljfn( pcjfn) THEN NULL;               06873
                err( $tstring );                                    06874
                END;                                                06875
             END;                                                   06876
       % now tell user we have set archive status for this file %
                                                                    06877
          *astr* _ *astr*, "   ", *jfnname*;                        06878
          IF pcjfn THEN *astr* _ *astr*, " and its partial copy";
                                                                    06879
          *astr* _ *astr*, CR, LF;                                  06880
       % now get rid of the jfn for the partial %                   06881
          IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                    06882
       % now get the next file in the group %                       06883
          (parcnxjfn):                                              06884
          r1.LH _ jfnflgs;                                          06885
          r1.RH _ fjfn;                                             06886
          IF NOT SKIP !gnjfn( r1 ) THEN EXIT LOOP;                  06887
       END;                                                         06888
% now get rid of any lingering jfns %                               06889
    IF NOT SKIP !rljfn( fjfn ) THEN NULL;                           06890
    IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                          06891
% all done now, so return %                                        06892
   RETURN;                                                          06893
END.   %%
```

```
                                                                    08695
%break%                                                              067
   (cbresta)   % GB: core NLS Break Statement procedure %
   PROCEDURE (bug, rlevcnt, tp1, tp2 % => statement %);              08207
      % Procedure description                                       08208
        FUNCTION                                                    08209
           Breaks a statement at the first non-printing character after
           the place designated by 'bug'.  Then deletes the following
           non-printing characters up to the first printing character.
           Inserts the new statement after the old one at the relative
           level specified by 'rlevcnt'.  A string may be specified to
           insert at the beginning of the new statement.  (This is
           symmetric with 'cappsta'.)                               08210
        ARGUMENTS                                                   08226
           bug - TEXT POINTER - character at which to break         08235
           rlevcnt - INTEGER - level of the new statement           08229
              -1 => down one level from old statement               08230
              0  => same level as old statement                     08231
              +n => up n levels from old statement                  08232
                 Global variables levdown (-1), levsuc (0) and levup
                 (+1) are recommended for this argument.            09585
           tp1 - TEXT POINTER - start of string to insert           08233
           tp2 - TEXT POINTER - end of string to insert             08234
        RESULTS                                                     08213
           procedure-value - STID - new statement made              08214
        NON-STANDARD CONTROL                                        08215
           none                                                     08216
        GLOBALS                                                     08217
           none                                                     08218
        EXAMPLE                                                     08219
           cbresta($tp1, levdown, $tp2, $tp3)                       08220
        %                                                           08221
      % Declarations %                                              08222
        LOCAL newstd; %new stid%                                     070
        REF bug, tp1, tp2;                                           071
       %msnst();%                                                    072
      newstd _ newrng(bug.stfile);                                   073
      insgrp(bug, rlevcnt, newstd, newstd);                          074
      FIND bug > CH $PT ^p1 $NP ^p2;                                 075
      cldtxt($p1, $p2);                                              076
      ST newstd _ tp1 tp2, p2 SE(bug);                               077
      ST bug _ SF(bug) p1;                                           078
       %msfst();%                                                    079
      RETURN(newstd);                                                080
      END.   %%
```

```
                                                                    08225
%connect to file directory%                                          094
   (cconfildir)   % GB: core NLS Connect to File Directory procedure %
   PROCEDURE (dir1, dir2, pass1, pass2 % => no value %);            08024
      % Procedure description                                       08025
        FUNCTION                                                    08026
           Connects to the directory specified, checking that the
           password matches the password specified.  The directory and
           password may be upper or lower case.                     08027
        ARGUMENTS                                                   08028
           dir1 - TEXT POINTER - start of directory name            08043
           dir2 - TEXT POINTER - end of directory name              08044
           pass1 - TEXT POINTER - start of password                 08045
           pass2 - TEXT POINTER - end of password                   08046
        RESULTS                                                     08030
           none                                                     08031
        NON-STANDARD CONTROL                                        08032
           Error if the directory does not exist or the password is
           incorrect.                                               08033
        GLOBALS                                                     08034
           CHANGED                                                  08035
              jpassw, ipassw                                        09376
           READ ONLY                                                09377
              chbmty                                                09378
        EXAMPLE                                                     08036
           cconfildir($tp1, $tp2, $tp3, $tp4)                       08037
        %                                                           08038
      % Declarations %                                              08039
        LOCAL                                                       07592
           dirnum,    % holds directory number %                    07593
           bytptr,    % byte pointer for use in STDIR and CNDIR jsies %
                                                                    07594
           recog;     % flag to do recognition or not %             07595
        LOCAL STRING                                                07597
           dirname[40],     % string that gets ASCIZ directory name %
                                                                    07598
           password[40];    % string that gets ASCIZ password name %
                                                                    07599
        REF dir1, dir2, pass1, pass2;                               07601
      % build needed upper case ASCIZ strings for jsies %           07603
        FIND dir1 > $(SP/TAB) ^dir1 dir2 < $(SP/TAB) ^dir2;         07604
        *dirname* _ + dir1 dir2, 0;                                 07605
        dirname.L _ dirname.L-1;                                    09892
        FIND pass1 > $(SP/TAB) ^pass1 pass2 < $(SP/TAB) ^pass2;     07606
        *password* _ + pass1 pass2, 0;                              07607
      % perform recognition if last char was a ^F, <ALT>, or EOL %  07608
        CASE *dirname*[dirname.L] OF                                07609
           =$ctlf, =$ascalt, =EOL:                                  07610
              BEGIN                                                 07611
              *dirname*[dirname.L] _ 0;        % null last byte %   07612
              recog _ -1;      % turn on recognition %            07613
              END;                                                  07614
           ENDCASE recog _ 0;  % no recognition %                   07615
      % get directory number given directory name; no recognition %
                                                                    07618
        IF NOT (dirnum _ stdircall($dirname, recog)) THEN           09889
```

```
        BEGIN                                                          09890
        % ambiguous or illegal directory name passed %                 07623
        (nodir): err($"no such directory");                            07624
        END;                                                           09891
% now connect; only error possible since we are already logged in
and we got a valid directory number above, is invalid passowrd %
                                                                       07629
   IF NOT condc(dirnum, 0, $password) THEN err($"incorrect
   password");                                                         07630
% success, so save password incase we call conjdir or conidir %
                                                                       07631
   *jpassw* _ *password*;                                              07632
   *ipassw* _ *password*;                                              07633
% all done so return %                                                07634
   RETURN;                                                             07635
END.   %%
```

```
                                                                        08042
%copy%                                                                   0143
   (ccoparcdir)   % UB: core NLS Copy Archive Directory procedure %
   PROCEDURE (stid, rlevcnt, tp1, tp2, params % => statement %);        08705
      % Procedure description                                         08706
         FUNCTION                                                     08707
            Not implemented yet.                                      08708
         ARGUMENTS                                                    08709
            none                                                      08710
         RESULTS                                                      08711
            procedure-value - STID - first new statement              08712
         NON-STANDARD CONTROL                                         08713
            none                                                      08714
         GLOBALS                                                      08715
            none                                                      08716
         EXAMPLE                                                      08717
            none                                                      08718
         %                                                            08719
      % Declarations %                                               08720
         REF tp1, tp2;                                                08726
      err( notyet );                                                  08724
      RETURN (stid); %stid of first new statement%                    08725
      END.   %%
```

```
                                                                    08723
(ccopdir)   % GB: core NLS Copy Directory procedure %
PROCEDURE (stid, rlev, info, gropk, sortk, rhstn, fname % =>
statement %);                                                       08047
   % Procedure description                                         08048
      FUNCTION                                                      08049
         Inserts after 'stid' a group of statements containing links
         to the files specified by 'fname'.  A variety of constraints
         can be placed on what is listed, how they are grouped, and
         how they are sorted.  However using anything but zero for
         the third, fourth and fifth arguments requires a certain
         amount of sophistication.  See the NLS system definitions of
         these records for more information; FEEDBACK will tell you
         where they are.                                            08050
      ARGUMENTS                                                     08051
         stid - STID - statement after which to insert the links
                                                                    08066
         rlev - INTEGER - level at which to insert them             09509
            -1 => down one level from 'stid'                        09510
            0  => same level as 'stid'                              09511
            +n => up n levels from 'stid'                           09512
               Global variables levdown (-1), levsuc (0) and levup
               (+1) are recommended for this argument.              09586
         info - DLINFO - record describing what to list, or 0       08068
         gropk - DLGRP - record containing grouping information, or 0
                                                                    08069
         sortk - DLSORT - record containing sorting information, or 0
                                                                    08070
         rhstn - INTEGER -                                          09288
            number of the host on which the files reside.  Only the
            local host ('lhostn') is currently implemented.         09289
         fname - STRING -                                           09496
            file group name string in TENEX format.  Connected
            directory is assumed if a directory is not specified.
            Examples:                                               09497
               "*.*"                                                09498
               "<SMITH>*.PC"                                        09499
               "<SMITH>FOO.NLS;3"                                   09500
      RESULTS                                                       08053
         procedure-value - STID -                                   08054
            first inserted statement, or 'stid' if no statements were
            inserted                                                09406
      NON-STANDARD CONTROL                                          08055
         Error if                                                   08056
            host is not the local host                              09407
            files are not disk files                                09408
            NLS storage problems                                    09410
            usual TENEX file system errors                          09409
      GLOBALS                                                       08057
         READ ONLY                                                  08058
            levup, levdown, levsuc, gtjoif, gtjstr, gtjidl, lhostn
                                                                    09411
      EXAMPLE                                                       08059
         ccopdir(stid, levdown, 0, 0, 0, lhostn, $"<smith>*.nls")
                                                                    08060
            inserts down one level from 'stid' a group of statements
```

```
        of the form                                                    09379
           < SMITH, FOO.NLS;7, >                                       09380
           < SMITH, BAZ.NLS;52, >  [ Being Modified By NAME
           (IDENT) ]                                                   09381
              etc.                                                     09391
   %                                                                   08061
% Declarations %                                                       08062
   LOCAL                                                               04939
      noverf,      % no version numbers flag %                        04940
      noextf,      % no extension names flag %                        04941
      sortdi,      % sort direction %                                 04942
      retstid,     % return stid %                                    04943
      tptr,        % temp pointer for building final string %         04944
      chns,        % pointer to data structure %                      04945
      adlmb,       % address of directory list storage %              04946
      fjfn,        % main jfn %                                       04947
      jfnflgs,     % left half flags for a group jfn %                04948
      flags;       % bits indicating * typed for file name fields %
                                                                       04949
   LOCAL TEXT POINTER                                                  04952
      strbgn, strend, tp1, tp2, tp3;                                   04953
   LOCAL STRING                                                        04954
      nwname[200],          % current file name link %                 04955
      oldname[200],         % previous file name link %                04956
      astr[2000],           % string for listing a file %              04957
      uname[200],           % complete name as entered by user %
                                                                       04958
      udirname[40],         % user input directory name %              04959
      ufilname[40],         % user input file name %                   04960
      uextname[40],         % user input extension name %              04961
      uverno[40],           % user input version number %              04962
      ulftovr[40],          % user input beyond version number %
                                                                       04963
      jfnname[200],         % complete name from jfns %                04964
      errstring[200],       % eror message string %                    04965
      tstring[200];         % temporary string %                       04966
   REF stid, tptr, chns, adlmb, fname;                                 04968
% initial variables %                                                  04970
   fjfn _ jfnflgs _ flags _ retstid _ &tptr _ &chns _ &adlmb _ 0;
                                                                       04971
   sortdi _ sortk.dlsrvr := FALSE;                                     04972
   noverf _ info.dlinvr := FALSE;                                      04973
   noextf _ info.dlinex := FALSE;                                      04974
% find out if remote or local (disk) file %                           04975
   CASE rhstn OF                                                       04976
      = lhostn:  NULL;                                                 04978
      ENDCASE err( $"Remote File Manipulations Not Implemented
      Yet" );                                                          04979
% now parse user input strings %                                      04980
   parseinput( &fname, $udirname, $ufilname, $uextname, $uverno,
   $ulftovr, $uname, $flags);                                          04981
% get main jfn (old file only, group jfn) %                           04982
   IF info.dlidlt THEN                                                 04983
      BEGIN                                                            04984
      IF NOT                                                           04985
         (fjfn _                                                       04986
```

```
            sgtjfn(                                                 04987
               gtjoif .V gtjstr .V gtjidl,                          04988
               $uname, $errstring : jfnflgs                         04989
            )                                                       04990
         )                                                          04991
            THEN err( $errstring );                                 04992
      END                                                           04993
   ELSE                                                             04994
      IF NOT                                                        04995
         (fjfn _                                                    04996
            sgtjfn(                                                 04997
               gtjoif .V gtjstr, $uname, $errstring : jfnflgs
                                                                    04998
            )                                                       04999
         )                                                          05000
            THEN err( $errstring );                                 05001
% now find out if we support this type of file %                    05002
   IF NOT chkdev( fjfn ) THEN err( $"only disk files supported" );
                                                                    05003
% get directory list storage %                                     05004
   IF NOT (&adlmb _ dlgmb(400)) THEN                                05005
      err($"Internal Storage Problem");                             05006
% cleanup on any problems %                                        05007
   ON SIGNAL ELSE                                                   05008
      BEGIN                                                         05009
      dlfmb(&adlmb := 0);                                           05010
      IF NOT SKIP !rljfn( fjfn ) THEN NULL;                         05011
      END;                                                          05012
% go create the data structure %                                   05013
   IF NOT  (&chns _                                                 05014
      dldriver(info, gropk, sortk, fjfn, jfnflgs, $errstring,
      &adlmb)) THEN                                                 05015
         BEGIN                                                      05016
         IF NOT SKIP !rljfn( fjfn ) THEN NULL;                      05017
         dlfmb( &adlmb := 0);                                       05018
         IF errstring.L THEN err($errstring)                        05019
         ELSE err($"System Screwup");                               05020
         END;                                                       05021
% broadcast any error message %                                    05022
   IF errstring.L THEN dismes(2,$errstring);                        05023
% build strings for one file, insert it, & loop for all files %
                                                                    05024
   WHILE &chns DO                                                   05025
      BEGIN                                                         05026
      IF (inptrf := FALSE) THEN EXIT LOOP;                          06768
      IF sortdi THEN &tptr _ chns.dlgbnu                            05027
      ELSE &tptr _ chns.dlgbsc;                                     05028
      WHILE &tptr DO                                                05029
         BEGIN                                                      05030
         IF (inptrf := FALSE) THEN EXIT LOOP 2;                     06769
         IF NOT tptr.dlfbfl.dlstig THEN                             05031
            BEGIN                                                   05032
            astr.L _ 0;                                             05033
            IF noverf OR noextf THEN                                05034
               BEGIN                                                05035
               FIND                                                 05036
```

```
                SE(*[tptr.dlfbal]*) [",] ^tp3 [";/".] ^tp2 [".]
                ^tp1;                                              05037
             IF noverf AND noextf THEN                             05038
                *nwname* _ SF(*[tptr.dlfbal]*) tp1,                05039
                   tp3 SE(*[tptr.dlfbal]*)                         05040
             ELSE                                                  05041
                IF noverf THEN                                     05042
                   *nwname* _ SF(*[tptr.dlfbal]*) tp2,             05043
                      tp3 SE(*[tptr.dlfbal]*)                      05044
                ELSE                                               05045
                   *nwname* _ SF(*[tptr.dlfbal]*) tp1,             05046
                      tp2 SE(*[tptr.dlfbal]*);                     05047
             IF *nwname* = *oldname* THEN GOTO cbypass             05048
             ELSE *oldname* _ *nwname*;                            05049
             *astr* _ *astr*, *nwname*;                            05050
             END                                                   05051
          ELSE                                                     05052
             *astr* _ *astr*, *[tptr.dlfbal]*;                     05053
          % build additional line 1 information string %           05054
             dlbldai(                                              05055
                fjfn, tptr.dlfbpf, tptr.dlfbff, tptr.dlfbpf,
                $astr);                                            05056
          % insert the header %                                    05057
             cnvcrlfteol($astr);  %convert CR LF to EOL for
             files%                                                09779
             FIND SF(*astr*) ^strbgn SE(*astr*) ^strend;           05058
             stid _ cinssta( stid, rlev, $strbgn, $strend);
                                                                   05059
             IF NOT retstid THEN retstid _ stid;                   05060
             rlev _ levsuc;                                        05061
             astr.L _ 0;                                           05062
          % build info string for this file & insert it %          05063
             dlbldi(                                               05064
                info, tptr.dlfbpf, tptr.dlfbff, tptr.dlfbpf,
                $"", $astr);                                       05065
             IF astr.L THEN                                        05066
                BEGIN                                              05067
                cnvcrlfteol($astr);  %convert CR LF to EOL for
                files%                                             09778
                FIND SF(*astr*) ^strbgn SE(*astr*) ^strend;
                                                                   05068
                rlev _ levdown;                                    05069
                stid _ cinssta( stid, rlev, $strbgn, $strend);
                                                                   05070
                rlev _ levup;                                      05071
                astr.L _ 0;                                        05072
                END;                                               05073
          END;                                                     05074
       (cbypass):                                                  05075
       IF sortdi THEN &tptr _ tptr.dlfbpb                          05076
       ELSE &tptr _ tptr.dlfbnb;                                   05077
       END;                                                        05078
    &chns _ chns.dlgbnb;                                           05079
    END;                                                           05080
% get rid of any lingering storage %                                05081
  dltmb( &adlmb := 0 );                                            05082
```

```
   % make sure we get rid of any jfns %                                  05083
      IF NOT SKIP !rljfn( fjfn ) THEN NULL;                             05084
   % return %                                                           05085
      RETURN( IF retstid THEN retstid ELSE stid );                      05086
   END.   %%
```

```
                                                                    08065
(ccopfil)   % GB: core NLS Copy File procedure %
PROCEDURE (rhost1, file1, rhost2, file2, astr % => no value %); 08005
   % Procedure description                                       08006
      FUNCTION                                                  08007
         Copies the NLS file(s) specified by 'file1' on host 'rhost1'
         to file(s) 'file2' on host 'rhost2'.  Any Partial Copies are
         also copied.  The destination files must already exist.
         Asterisks (*) in the source file specification (file1) must
         be matched by asterisks in the destination file
         specification (file2).  Messages describing the results are
         put in 'astr'.  Note: this copies the source file's origin
         statement along with its other statements.             08008
      ARGUMENTS                                                 08009
         rhost1 - INTEGER -                                     09420
            number of the host on which the files reside.  Only the
            local host ('lhostn') is currently implemented.     09421
         file1 - STRING -                                       09518
            file group name in TENEX format.  Connected directory is
            assumed if a directory is not specified.  Examples: 09519
               "*.*"                                            09520
               "<SMITH>*.PC"                                    09521
               "<SMITH>FOO.NLS;3"                               09522
         rhost2 - INTEGER -                                     09426
            number of the host to which the files are to be copied.
            Only the local host ('lhostn') is currently implemented.
                                                                09427
         file2 - STRING -                                       09523
            file group name in TENEX format.  Connected directory is
            assumed if a directory is not specified.  Examples: 09524
               "*.*"                                            09525
               "<SMITH>*.PC"                                    09526
               "<SMITH>FOO.NLS;3"                               09527
            Note: the *'s here must match one-for-one with the *'s in
            file1.                                              09627
         astr - STRING - string in which to put messages        09432
      RESULTS                                                   08011
         none                                                   08014
      NON-STANDARD CONTROL                                      08013
         Error if                                               08012
            host is not local host                              09528
            illegal use of <CTRL-F>, $ or * (see the TENEX manual for
            the correct use of these characters)                09529
      GLOBALS                                                   08015
         gtjoif, gtjstr, lhostn - read only                     08016
      EXAMPLE                                                   08017
         ccopfil(lhostn, $"<smith>*.nls", lhostn, $"<jones>*.nls",
         $string)                                               08018
      %                                                         08019
   % Declarations %                                            08020
      LOCAL                                                     06069
         jfn1,          % main jfn %                          06070
         jfn2,          % jfn for destination file %          06071
         jfnfl1,        % left half flags for a group jfn %   06072
         jfnfl2,        % left half flags for destination group jfn %
                                                                06073
```

```
        pcjfn1,     % jfn of partial copy %                          06074
        pcjfn2,     % jfn of partial copy %                          06075
        flag1,      % bits indicating * typed for file name fields %
                                                                     06076
        flag2;      % * typed for destination file name fields %
                                                                     06077
    LOCAL STRING                                                     06079
        uf1[200],   % complete name as entered by user %             06080
        jfnnm1[200],% complete name from jfns %                      06081
        uf2[200],   % complete destination name as entered by user %
                                                                     06082
        udir2[40],  % user input directory destination name %        06083
        ufil2[40],  % user input file destination name %             06084
        uext2[40],  % user input extension destination name %        06085
        uver2[40],  % user input destination version number %        06086
        ulft2[40],  % user destination input beyond version number %
                                                                     06087
        jfnnm2[200],        % complete destination name from jfns %
                                                                     06088
        errstring[200],     % eror message string %                  06089
        tstring[200];       % temporary string %                     06090
    REF file1, file2, astr;                                          06091
% initial variables %                                               06093
    jfn1 _ jfnfl1 _ pcjfn1 _ flag1 _ 0;                              06094
    jfn2 _ jfnfl2 _ pcjfn2 _ flag2 _ 0;                              06095
% find out if remote or local (disk) file %                        06096
    CASE rhost1 OF                                                   06097
        = lhostn:  NULL;                                             06098
        ENDCASE err( $"remote file manipulations not implemented
        yet" );                                                      06099
    CASE rhost2 OF                                                   06100
        = lhostn:  NULL;                                             06101
        ENDCASE err( $"remote file manipulations not implemented
        yet" );                                                      06102
% now parse user source input string %                             06103
    parseinput( &file1, $udir2, $ufil2, $uext2, $uver2, $ulft2,
    $uf1, $flag1);                                                   06104
% parse user destination input string %                            06105
    parseinput( &file2, $udir2, $ufil2, $uext2, $uver2, $ulft2,
    $uf2, $flag2);                                                   06106
    % illegal to terminate destination fields with ^F or alt %
                                                                     06107
        CASE *udir2*[udir2.L] OF                                     06108
            = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
                                                                     06109
            ENDCASE;                                                 06110
        CASE *ufil2*[ufil2.L] OF                                     06111
            = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
                                                                     06112
            ENDCASE;                                                 06113
        CASE *uext2*[uext2.L] OF                                     06114
            = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
                                                                     06115
            ENDCASE;                                                 06116
        CASE *uver2*[uver2.L] OF                                     06117
            = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
```

```
                                                                    06118
        ENDCASE;                                                    06119
      CASE *ulft2*[ulft2.L] OF                                      06120
        = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
                                                                    06121
        ENDCASE;                                                    06122
% check for matching astericks %                                    06123
   IF flag1.dstar AND NOT flag2.dstar THEN err($"Illegal use of
   *");                                                             06124
   IF flag1.fstar AND NOT flag2.fstar THEN err($"Illegal use of
   *");                                                             06125
   IF flag1.estar AND NOT flag2.estar THEN err($"Illegal use of
   *");                                                             06126
   IF flag1.vstar AND NOT flag2.vstar THEN err($"Illegal use of
   *");                                                             06127
% get main source jfn (old file only, group jfn) %                 06128
   IF NOT (jfn1 _ sgtjfn( gtjoif .V gtjstr + 0, $uf1, $errstring :
   jfnf11 ) )THEN err( $errstring );                                06129
% now find out if we support this type of file %                   06130
   IF NOT chkdev( jfn1 ) THEN err( $"only disk files supported" );
                                                                    06131
% now loop to get all files in the group %                         06132
   LOOP                                                             06133
      BEGIN                                                         06134
      % check to make sure this not a PC with * for extension name
      %                                                             06135
         IF NOT chkpcs( jfn1, flag1) THEN GOTO gtcnxtjfn;           06136
      % get actual file name into string for use later (maybe) %
                                                                    06137
         r3 _ 01111086    %  directory file ext ver %              06138
            + (IF jfnf11.lhjb9  THEN 186 ELSE 0) % ;Protection %
                                                                    06139
            + (IF jfnf11.lhjb10 THEN 185 ELSE 0) % ;Account %
                                                                    06140
            + (IF jfnf11.lhjb11 THEN 484 ELSE 0) % ;T %             06141
            + 1;          % with proper file punctuation %         06142
         jfnflink( jfn1, $jfnnm1, r3);                              06143
      % get source PC jfn & find if this user can access this PC %
                                                                    06144
         IF NOT getpcjfn( jfn1, $pcjfn1, $errstring, FALSE) THEN
                                                                    06145
            BEGIN % this user can't access this file %              06146
            *tstring* _ *jfnnm1*, " not copied because [",
            *errstring*, ']';                                       06147
            IF flag1 THEN                                           06148
               BEGIN                                                06149
               *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
                                                                    06150
               GOTO gtcnxtjfn;                                      06151
               END                                                  06152
            ELSE                                                    06153
               BEGIN                                                06154
               IF NOT SKIP !rljfn( jfn1) THEN NULL;                 06155
               err( $tstring );                                     06156
               END;                                                 06157
            END;                                                    06158
```

```
% get destination jfns %                                          06159
  IF NOT jfn2 _                                                  06160
     gtdesjfn( jfn1, pcjfn1, $udir2, $ufil2, $uext2,
     $uver2, $ulft2, flag2, $errstring : pcjfn2) THEN            06161
        BEGIN      % can't copy file (or partial copy) %
                                                                 06162
        *tstring* _ *jfnnm1*, " not copied because ",
        *errstring*;                                             06163
        IF flag1 THEN                                            06164
           BEGIN                                                 06165
           *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
                                                                 06166
           GOTO gtcnxtjfn;                                       06167
           END                                                   06168
        ELSE                                                     06169
           BEGIN                                                 06170
           IF NOT SKIP !rljfn( jfn1) THEN NULL;                  06171
           IF NOT SKIP !rljfn( pcjfn1) THEN NULL;                06172
           err( $tstring );                                      06173
           END;                                                  06174
        END;                                                     06175
% get destination file name into string for use later %
                                                                 06176
  r3 _ 011110B6    %  directory file ext ver %                   06177
     + (IF jfnfl1.lhjb9  THEN 1B6 ELSE 0) % ;Protection %
                                                                 06178
     + (IF jfnfl1.lhjb10 THEN 1B5 ELSE 0) % ;Account %
                                                                 06179
     + (IF jfnfl1.lhjb11 THEN 4B4 ELSE 0) % ;T %                 06180
     + 1;          % with proper file punctuation %             06181
  jfnflink( jfn2, $jfnnm2, r3);                                  06182
% now move the file (and its partial copy) %                     06183
  IF NOT copflandpc(jfn1, pcjfn1, jfn2, pcjfn2, $errstring)
  THEN                                                           06184
     BEGIN % can't move file (or partial copy) %                 06185
     *tstring* _ *jfnnm1*, " not copied because ",
     *errstring*;                                                06186
     IF flag1 THEN                                               06187
        BEGIN                                                    06188
        *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
                                                                 06189
        GOTO gtcnxtjfn;                                          06190
        END                                                      06191
     ELSE                                                        06192
        BEGIN                                                    06193
        IF NOT SKIP !rljfn( jfn1) THEN NULL;                     06194
        IF NOT SKIP !rljfn( pcjfn1) THEN NULL;                   06195
        err( $tstring );                                         06196
        END;                                                     06197
     END;                                                        06198
% now tell the user we have copied this file %                   06199
  *astr* _ *astr*, "   ", *jfnnm1*;                              06200
  IF pcjfn1 THEN *astr* _ *astr*, " [and PC]";                   06201
  *astr* _ *astr*, " to ", *jfnnm2*;                             06202
  IF pcjfn2 THEN *astr* _ *astr*, " [and PC]";                   06203
  *astr* _ *astr*, CR, LF;                                       06204
```

```
        % now get rid of the jfn for the partial %                    06205
           IF NOT SKIP !rljfn( pcjfn1 ) THEN NULL;                    06206
        % now get the next file in the group %                        06207
           (gtcnxtjfn):                                               06208
           r1.LH _ jfnfl1;                                            06209
           r1.RH _ jfn1;                                              06210
           IF NOT SKIP !gnjfn( r1 ) THEN EXIT LOOP;                   06211
        END;                                                          06212
  % now get rid of any lingering jfns %                               06213
     IF NOT SKIP !rljfn( jfn1 ) THEN NULL;                            06214
     IF NOT SKIP !rljfn( pcjfn1 ) THEN NULL;                          06215
  % all done now, so return %                                         06216
    RETURN;                                                           06217
  END.   %%
```

```
                                                                    08023
(ccopgro)   % GB: core NLS Copy Group procedure %
PROCEDURE (tostid, rlevcnt, fromstid1, fromstid2, filter, vsptr % =>
statement, statement %);                                            08301
   % Procedure description                                          08302
      FUNCTION                                                      08303
         Copies the group at 'fromstid1' through 'fromstid2' to
         follow 'tostid' at the relative level 'rlevcnt'.  If
         'filter' is TRUE, copies only those statements which pass
         the viewspecs in 'vsptr'.                                  08304
      ARGUMENTS                                                     08305
         tostid - STID -                                            08306
            statement after which to insert the new group           09438
         rlevcnt - INTEGER - level of the new group                 09444
            -1 => down one level from 'tostid'                      09445
            0  => same level as 'tostid'                            09446
            +n => up n levels from 'tostid'                         09447
               Global variables levdown (-1), levsuc (0) and levup
               (+1) are recommended for this argument.              09587
         fromstid1 - STID - head of group to copy                   09440
         fromstid2 - STID - tail of group to copy                   09441
         filter - BOOLEAN -                                         09442
            TRUE  => filter statements                              09448
            FALSE => copy all statements                            09449
         vsptr - VIEWSPECS -                                        09443
            viewspecs through which to filter statements, or 0.     09450
      RESULTS                                                       08307
         procedure-value - STID - head of new group                 08308
         2nd result - STID - tail of new group                      09437
      NON-STANDARD CONTROL                                          08309
         none                                                       08310
      GLOBALS                                                       08311
         none                                                       08312
      EXAMPLE                                                       08313
         head _                                                     08314
            ccopgro(stid1, levdown, stid2, stid3, FALSE, 0 : tail)
                                                                    09648
         head _                                                     09451
            ccopgro(stid1, 3*levup, stid2, stid3, TRUE, $vs : tail)
                                                                    09649
      %                                                             08315
   % Declarations %                                                08316
      LOCAL newhead, newtail;                                       06407
      REF vsptr;                                                    06408
   fromstid1 _ grptst(fromstid1, fromstid2 : fromstid2);            09692
   IF NOT filter THEN                                               09693
      newhead _ copgrp(tostid, rlevcnt, fromstid1, fromstid2, FALSE:
      newtail)                                                      09694
   ELSE                                                             09695
      newhead _ copfgrp(tostid, rlevcnt, fromstid1, fromstid2,
      &vsptr: newtail);                                             09696
   RETURN(newhead, newtail);                                        09697
   END.
                                                                    09698
(ccopseqfil)   % GB: core NLS Copy Sequential File procedure %
PROCEDURE (stid, rlevcnt, tp1, tp2, filtyp % => statement %);       08727
```

```
% Procedure description                                                08728
   FUNCTION                                                           08729
      Copies a sequential file (e.g. TENEX character file) into
      NLS structured form.  Inserts it to follow 'stid' at the
      relative level 'rlevcnt'.  The structure that is built
      depends on 'filtyp' as follows:                                 08730
         *** ASK HARVEY ***                                           09452
   ARGUMENTS                                                          08731
      stid - STID -                                                   09457
         statement after which to insert the new statements           09458
      rlevcnt - INTEGER - level of the new statements                 09459
         -1 => down one level from 'stid'                             09460
         0  => same level as 'stid'                                   09461
         +n => up n levels from 'stid'                                09462
            Global variables levdown (-1), levsuc (0) and levup
            (+1) are recommended for this argument.                   09588
      tp1 - TEXT POINTER - start of file name                         09463
      tp2 - TEXT POINTER - end of file name                           09464
      filtyp - INTEGER - *** ASK HARVEY ***                           09456
   RESULTS                                                            08733
      procedure-value - STID - first new statement                    08734
   NON-STANDARD CONTROL                                               08735
      none                                                            08736
   GLOBALS                                                            08737
      heurfil - read only - *** ASK HARVEY ***                        08738
      justfil - read only - *** ASK HARVEY ***                        09465
   EXAMPLE                                                            08739
      *filename* _ "<smith>foo.nls";                                  08740
      FIND SF(*filename*) ^tp1 SE(*filename*) ^tp2;                   09466
      ccopseqfil(stid, levdown, $tp1, $tp2, justfil);                 09467
   %                                                                  08741
% Declarations %                                                       08742
   LOCAL STRING string[100];                                           0171
   REF tp1, tp2;                                                      05250
*string* _ tp1 tp2; %file name%                                       08746
CASE filtyp OF                                                        08747
   = heurfil: stid _ <SEQFIL, inseqn> (stid, rlevcnt, $string, 0);
                                                                      08748
   = justfil: stid _ <SEQFIL, inseqn> (stid, rlevcnt, $string, 1);
                                                                      08749
   ENDCASE                                                            08750
      stid _ <SEQFIL, inseq> (stid, rlevcnt, $string, filtyp);
                                                                      08751
RETURN (stid); %stid of first new statement%                          08752
END.   %%
```

```
                                                                    08745
(ccopsta)   % GB: core NLS Copy Statement procedure %
PROCEDURE (tostid, rlevcnt, fromstid, filter, vsptr % => statement
%);                                                                 08257
   % Procedure description                                         08258
      FUNCTION                                                     08259
         Copies the statement at 'fromstid' to follow 'tostid' at the
         relative level 'rlevcnt'.  If 'filter' is TRUE, copies only
         those statements which pass the viewspecs in 'vsptr'.     09468
      ARGUMENTS                                                    08261
         tostid - STID -                                           09469
            statement after which to insert the new statement      09470
         rlevcnt - INTEGER - level of the new group                09471
            -1 => down one level from 'tostid'                     09472
            0  => same level as 'tostid'                           09473
            +n => up n levels from 'tostid'                        09474
               Global variables levdown (-1), levsuc (0) and levup
               (+1) are recommended for this argument.             09589
         fromstid - STID - statement to copy                       09475
         filter - BOOLEAN -                                        09477
            TRUE  => filter statements                             09478
            FALSE => copy all statements                           09479
         vsptr - VIEWSPECS -                                       09480
            viewspecs through which to filter statements, or 0.    09481
      RESULTS                                                      08263
         procedure-value - STID - new statement made               08264
      NON-STANDARD CONTROL                                         08265
         none                                                      08266
      GLOBALS                                                      08267
         none                                                      08268
      EXAMPLE                                                      08269
         ccopsta(stid1, levdown, stid2, FALSE, 0)                  08270
      %                                                            08271
   % Declarations %                                               08272
      LOCAL                                                        07442
         da, savvs1, savvs2,                                       07443
         newsrc; %new stid%                                        07444
      LOCAL STRING locstr[10];                                     07445
      REF vsptr, da;                                               07446
   IF NOT filter THEN                                              08276
      BEGIN                                                        08277
      newsrc _ newrng(tostid.stfile);                              08278
      insgrp(tostid, rlevcnt, newsrc, newsrc);                     08279
      copplist(fromstid, newsrc);                                  08280
      END                                                          08281
   ELSE                                                            08282
      BEGIN                                                        08283
      % set viewspecs to get only one statement %                  08284
         *locstr* _ "eg";                                          08285
         &da _ lda();                                              08286
         savvs1 _ da.davspec := vsptr.vs1;                         08287
         savvs2 _ da.davspc2 := vsptr.vs2;                         08288
         ON SIGNAL ELSE                                            08289
            BEGIN                                                  08290
            da.davspec _ savvs1;                                   08291
            da.davspc2 _ savvs2;                                   08292
```

```
          END;                                                    08293
        feedlt( &da, $locstr );                                   08294
        vsptr.vs1 _ da.davspec := savvs1;                         08295
        vsptr.vs2 _ da.davspc2 := savvs2;                         08296
        ON SIGNAL ELSE;                                           08297
      newsrc _ copfgrp(tostid, rlevcnt, fromstid, fromstid, &vsptr);
                                                                  08298
    END;                                                          08299
  RETURN(newsrc);                                                 08300
  END.   %%
```

```
                                                                    08275
(ccoptex)   % GB: core NLS Copy Text procedure %
PROCEDURE (totptr, fromtptr1, fromtptr2, insertspace % => no value
%);                                                                 08623
   % Procedure description                                         08624
      FUNCTION                                                     08625
         Copies the text at 'fromtptr1' through 'fromtptr2' to follow
         the character designated by 'totptr'.  If 'insertspace' is
         TRUE, then a space is inserted immediately after totptr
         (i.e. before the inserted text).                          08626
      ARGUMENTS                                                    08627
         totptr - TEXT POINTER -                                   09490
            character after which to insert the new text           09494
         fromtptr1 - TEXT POINTER - start of text to copy          09491
         fromtptr2 - TEXT POINTER - end of text to copy            09492
         insertspace - BOOLEAN - TRUE => insert a space first      09493
      RESULTS                                                      08629
         none                                                      08630
      NON-STANDARD CONTROL                                         08631
         none                                                      08632
      GLOBALS                                                      08633
         none                                                      08634
      EXAMPLE                                                      08635
         ccoptex($tp1, $tp2, $tp3, FALSE)                          08636
      %                                                            08637
   % Declarations %                                               08638
      REF totptr, fromtptr1, fromtptr2;                             0189
     %msntx();%                                                    08643
   IF insertspace THEN                                           08644
      ST totptr _ SF(totptr) totptr, " ",                        08645
         $fromtptr1 fromtptr2, % $ => dont move markers %        08646
         totptr SE(totptr)                                       08647
   ELSE                                                          08648
      ST totptr _ SF(totptr) totptr,                             08649
         $fromtptr1 fromtptr2, % $ => dont move markers %        08650
         totptr SE(totptr);                                      08651
     %msftx();%                                                  08652
   RETURN;                                                       08653
   END.   %%
```

```
                                                                    08641
%create file%                                                        0202
   (ccrefil)   % GB: core NLS Create File procedure %
   PROCEDURE (rhostn, fname % => statement %);                      09730
      % Procedure description                                      09731
         FUNCTION                                                 09732
            Creates an NLS file having name 'fname' on host 'rhostn'.
                                                                    09733
         ARGUMENTS                                                09734
            hostn - INTEGER -                                     09735
               number of the host on which the file is to reside.  Only
               the local host ('lhostn') is currently implemented. 09736
            fname - STRING -                                      09737
               file name in TENEX format.  ".NLS" is assumed if an
               extension is not specified.  Connected directory is
               assumed if a directory is not specified.           09738
         RESULTS                                                  09739
            procedure-value - STID - origin statement of created file
                                                                    09740
         NON-STANDARD CONTROL                                     09741
            none                                                  09742
         GLOBALS                                                  09743
            read, origff, oldvrsn, origin, oldversion - read only 09744
         EXAMPLE                                                  09745
            ccrefil(lhostn, $"<SMITH>FOO.NLS")                    09746
            ccrefil(lhostn, $"BAZ")                               09747
         %                                                        09748
      % Declarations %                                           09749
         LOCAL stid, jfn, flgbits, errcode, fileno;               09750
         REF fname;                                               09751
      CASE rhostn OF                                              09752
         = lhostn:                                                09753
            BEGIN                                                 09754
            IF NOT FIND SF(*fname*) ['.] THEN *fname* _ *fname*, ".NLS";
                                                                    09755
            flgbits _ getgtjflg(read, origff, oldvrsn);           09756
            IF NOT (jfn _ sgtjfn(flgbits, &fname, 0 : errcode)) THEN
                                                                    09757
               BEGIN                                              09758
               CASE errcode OF                                    09759
                  =$gjfx24, =$gjfx20, =$gjfx19, =$gjfx18: NULL;   09760
                     %expect "old file required" or "no such version" or
                     "no such extension" or "no such filename"%   09761
                  ENDCASE %got something else%                    09762
                     BEGIN                                        09763
                     gtjerr(errcode, $lit, flgbits);              09764
                     err($lit);                                   09765
                     END;                                         09766
               END                                                09767
            ELSE  %already have a file by that name%              09768
               err($"A file by that name already exists!");       09769
            fileno _ openull($fname);                             09770
            stid _ origin;                                        09771
            stid.stfile _ fileno;                                 09772
            updtfl(fileno, oldversion, 0);  %update to old%       09773
            RETURN(stid);  %stid of origin%                       09774
```

```
        END;                                                    09775
    ENDCASE err($"Remote File Manipulations Not Implemented Yet");
                                                                09776
END.    %%
```

```
                                                                        09777
%current location%                                                       0231
   (ccurloc)   % UB: core NLS Current Location in file procedure %
   PROCEDURE (tptr, string % => no value %);                            08753
      % Procedure description                                          08754
         FUNCTION                                                       08755
            none                                                        08756
         ARGUMENTS                                                      08757
            none                                                        08758
         RESULTS                                                        08759
            none                                                        08760
         NON-STANDARD CONTROL                                           08761
            none                                                        08762
         GLOBALS                                                        08763
            none                                                        08764
         EXAMPLE                                                        08765
            none                                                        08766
         %                                                              08767
      % Declarations %                                                 08768
         REF tptr, string;                                              0234
      IF tda.davspec.vssidf                                           08772
         THEN % use SID's %                                           08773
            *string* _ " = 0", STRING( getsid( tptr ) )               08774
         ELSE % use statement number %                                08775
            BEGIN                                                       08776
            *string* _ " = ";                                           08777
            fechno(tptr, &string);                                      08778
            END;                                                        08779
      *string* _ *string*, " +", STRING(tptr[1]);                      08780
      RETURN;                                                          08781
      END.   %%
```

```
                                                                        08771
%current context%                                                        0246
   (ccurcon)   % UB: core NLS Current Context in file procedure %
   PROCEDURE (tptr, string % => no value %);                            08782
      % Procedure description                                          08783
         FUNCTION                                                       08784
            none                                                        08785
         ARGUMENTS                                                      08786
            none                                                        08787
         RESULTS                                                        08788
            none                                                        08789
         NON-STANDARD CONTROL                                           08790
            none                                                        08791
         GLOBALS                                                        08792
            none                                                        08793
         EXAMPLE                                                        08794
            none                                                        08795
         %                                                              08796
      % Declarations %                                                 08797
         LOCAL TEXT POINTER tcm, ptr;                                    0249
         REF tptr, string;                                               0250
      tcm _ ptr _ tptr;                                                 08801
      tcm[1] _ tptr[1];                                                 08802
      ptr[1] _ MAX(1, tcm[1]-tslshchars);                               08803
      *string* _ SP, ptr tcm, LF, "==>";                                08804
      FIND SE(tcm) ^ptr;                                                08805
      ptr[1] _ MIN(tcm[1]+tslshchars+1, ptr[1]);                        08806
      *string* _ *string*, tcm ptr, "  ";                              08807
      RETURN;                                                           08808
      END.   %%
```

```
                                                                        08800
%delete%                                                                 0260
   (cdelallmar)   % GB: core NLS Delete All Markers procedure %
   PROCEDURE (fileno % => no value %);                                  08809
      % Procedure description                                          08810
         FUNCTION                                                       08811
            Deletes all markers from the file designated.  Markers are
            stored in the file header, so only the file header page is
            modified.                                                   08812
         ARGUMENTS                                                      08813
            fileno - INTEGER - NLS file number of the file              08814
         RESULTS                                                        08815
            none                                                        08816
         NON-STANDARD CONTROL                                           08817
            none                                                        08818
         GLOBALS                                                        08819
            origin, rngtyp, mkrtbl, filhed - read only                  08820
         EXAMPLE                                                        08821
            cdelallmar(stid.stfile)                                     08822
         %                                                              08823
      % Declarations %                                                  08824
         LOCAL mk, stid, aring;                                          0263
         REF aring;                                                     06767
      % make sure we have a locked file %                               08828
         stid _ origin;                                                 08829
         stid.stfile _ fileno;                                          08830
         lodent( stid, rngtyp : &aring);                                08831
         aring.rsub _ aring.rsub;                                       08832
      mk _ filhdr(fileno) + $mkrtbl - $filhed;                          08833
      [mk] _ 0;                                                         08834
      RETURN;                                                           08835
      END.   %%
```

```
                                                                    08827
%(cdelarcfil)   %% UB: core NLS Delete Archived File procedure %%
PROCEDURE (f1, f2 %% => no value %%);                               09290
   %% Procedure description                                        09291
      FUNCTION                                                      09292
         Not implemented yet.  Call commented out in PSEDIT.        09293
      ARGUMENTS                                                     09294
         none                                                       09295
      RESULTS                                                       09296
         none                                                       09297
      NON-STANDARD CONTROL                                          09298
         none                                                       09299
      GLOBALS                                                       09300
         none                                                       09301
      EXAMPLE                                                       09302
         none                                                       09303
      %%                                                            09304
   %% Declarations %%                                              09305
   err( notyet );                                                   09306
   RETURN;                                                          09307
   END.%
                                                                    09308
(cdelfil)   % GB: deletes a group of files and their partial copies %
PROCEDURE (rhstn, fname, astr % => no value %);                     07964
   % Procedure description                                         07965
      FUNCTION                                                      07966
         Deletes the NLS file(s) specified by 'fname' on host
         'rhstn'.  Any Partial Copies are also deleted.  Messages
         describing the results are put in 'astr'.                  09629
      ARGUMENTS                                                     07968
         rhstn - INTEGER -                                          09537
            number of the host on which the files reside.  Only the
            local host ('lhostn') is currently implemented.         09538
         fname - STRING -                                           09539
            file group name string in TENEX format.  Connected
            directory is assumed if a directory is not specified.
            Examples:                                               09540
               "*.*"                                                09541
               "<SMITH>*.PC"                                        09542
               "<SMITH>FOO.NLS;3"                                   09543
         astr - STRING - string to receive list of deleted files
                                                                    07985
      RESULTS                                                       07970
         none                                                       07971
      NON-STANDARD CONTROL                                          07972
         Error if                                                   07973
            host is not local host                                  09544
            file not disk file                                      09545
            user not allowed access to file and can't release JFN
                                                                    09546
            can't access or delete file or PC                       09547
      GLOBALS                                                       07974
         <lots>                                                     07975
      EXAMPLE                                                       07976
         cdelfil(lhostn, $"*.NLS", $string)                         07977
         cdelfil(lhostn, $"<SMITH>FOO.*;*" $string)                 09548
```

```
    %                                                                07978
  % Declarations %                                                   07979
    LOCAL                                                           01573
       fjfn,        % main jfn %                                    01574
       jfnflgs,     % left half flags for a group jfn %             01575
       pcjfn,       % jfn of partial copy %                         01576
       flags;       % bits indicating * typed for file name fields %
                                                                    01577
    LOCAL STRING                                                    01580
       uname[200],          % complete name as entered by user %
                                                                    01581
       udirname[40],        % user input directory name %           01582
       ufilname[40],        % user input file name %                01583
       uextname[40],        % user input extension name %           01584
       uverno[40],          % user input version number %           01585
       ulftovr[40],         % user input beyond version number %
                                                                    01586
       jfnname[200],        % complete name from jfns %             01587
       errstring[200],      % eror message string %                 01588
       tstring[200];        % temporary string %                    01589
    REF fname, astr;                                                01591
  % initial variables %                                             01593
    fjfn _ jfnflgs _ pcjfn _ flags _ 0;                             01594
  % find out if remote or local (disk) file %                       01595
    CASE rhstn OF                                                   01596
       = lhostn:  NULL;                                             01598
       ENDCASE err( $"remote file manipulations not implemented
       yet" );                                                      01599
  % now parse user input strings %                                  01600
    parseinput( &fname, $udirname, $ufilname, $uextname, $uverno,
    $ulftovr, $uname, $flags);                                      01601
  % get main jfn (old file only, group jfn) %                      01602
    IF NOT (fjfn _ sgtjfn( gtjoif .V gtjstr + 0, $uname, $errstring
    : jfnflgs ) )THEN err( $errstring );                            01603
  % now find out if we support this type of file %                  01604
    IF NOT chkdev( fjfn ) THEN err( $"only disk files supported" );
                                                                    01605
  % now loop to get all files in the group %                       01606
    LOOP                                                            01607
       BEGIN                                                        01608
       % check to make sure this not a PC with * for extension name
       %                                                            01609
          IF NOT chkpcs( fjfn, flags) THEN GOTO getnxtjfn;          01610
       % get actual file name into string for use later (maybe) %
                                                                    01611
          r3 _ 011110B6    %  directory file ext ver %             01612
             + (IF jfnflgs.lhjb9  THEN 1B6 ELSE 0) % ;Protection %
                                                                    01613
             + (IF jfnflgs.lhjb10 THEN 1B5 ELSE 0) % ;Account %
                                                                    01614
             + (IF jfnflgs.lhjb11 THEN 4B4 ELSE 0) % ;T %           01615
             + 1;           % with proper file punctuation %       01616
          jfnflink( fjfn, $jfnname, r3);                            01617
       % get PC jfn and find out if this user can access this PC %
                                                                    01618
          IF NOT getpcjfn( fjfn, $pcjfn, $errstring, FALSE) THEN
```

```
                                                                    01619
            BEGIN % this user can't access this file %              01620
            *tstring* _ *jfnname*, " not deleted because [",
            *errstring*, "];                                        01621
            IF flags THEN                                           01622
               BEGIN                                                01623
               *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
                                                                    01624
               GOTO getnxtjfn;                                      01625
               END                                                  01626
            ELSE                                                    01627
               BEGIN                                                01628
               IF NOT SKIP !rljfn( fjfn) THEN NULL;                 01629
               err( $tstring );                                     01630
               END;                                                 01631
            END;                                                    01632
         % now delete the file (and its partial copy) %             01633
           IF NOT delflandpc( fjfn, pcjfn, $errstring) THEN         01634
            BEGIN % can't delete file (or partial copy) %           01635
            *tstring* _ *jfnname*, " not deleted because ",
            *errstring*;                                            01636
            IF flags THEN                                           01637
               BEGIN                                                01638
               *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
                                                                    01639
               GOTO getnxtjfn;                                      01640
               END                                                  01641
            ELSE                                                    01642
               BEGIN                                                01643
               IF NOT SKIP !rljfn( fjfn) THEN NULL;                 01644
               IF NOT SKIP !rljfn( pcjfn) THEN NULL;                01645
               err( $tstring );                                     01646
               END;                                                 01647
            END;                                                    01648
         % now tell the user we have deleted this file %            01649
           *astr* _ *astr*, "   ", *jfnname*;                       01650
           IF pcjfn THEN *astr* _ *astr*, " and its partial copy";
                                                                    01651
           *astr* _ *astr*, CR, LF;                                 01652
         % now get rid of the jfn for the partial %                 01653
           IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                   01654
         % now get the next file in the group %                     01655
           (getnxtjfn):                                             01656
           r1.LH _ jfnflgs;                                         01657
           r1.RH _ fjfn;                                            01658
           IF NOT SKIP !gnjfn( r1 ) THEN EXIT LOOP;                 01659
         END;                                                       01660
      % now get rid of any lingering jfns %                        01661
        IF NOT SKIP !rljfn( fjfn ) THEN NULL;                       01662
        IF NOT SKIP !rljfn( pcjfn ) THEN NULL;                      01663
      % all done now, so return %                                  01664
       RETURN;                                                      01665
      END.   %%
```

```
                                                                    07982
(cdelgro)   % GB: Core NLS Delete Group Command %
PROCEDURE (stid1, stid2, filter, vsptr % => no value %);            08115
   % Procedure description                                          08116
      FUNCTION                                                      08117
         Deletes the group at 'stid1' through 'stid2'.  If 'filter'
         is TRUE, deletes only those statements which pass the
         viewspecs in 'vsptr'.                                      09549
      ARGUMENTS                                                     08119
         stid1 - STID - head of group to delete                     09556
         stid2 - STID - tail of group to delete                     09557
         filter - BOOLEAN -                                         09558
            TRUE  => filter statements                              09559
            FALSE => delete all statements                          09560
         vsptr - VIEWSPECS -                                        09561
            viewspecs through which to filter statements, or 0.     09562
      RESULTS                                                       08121
         none                                                       08122
      NON-STANDARD CONTROL                                          08123
         Error if *** ASK HARVEY ***                                08124
      GLOBALS                                                       08125
         endfil, deltflag - read only                               08126
      EXAMPLE                                                       08127
         cdelgro(stid1, stid2, FALSE, 0)                            08128
      %                                                             08129
   % Declarations %                                                 08130
      LOCAL newstid;                                                06459
      REF vsptr;                                                    06460
   stid1 _ grptst(stid1, stid2 :stid2);                             08134
   IF (newstid _ getnxt(getend(stid2))) = endfil THEN newstid _
   getbck(stid1);                                                   08135
   IF NOT filter THEN                                               08136
      BEGIN                                                         08137
      IF NOT remgrp(stid1, stid2) THEN                              08138
         err($"illegal delete");                                    08139
      delgrp(stid1, stid2, newstid);                                08140
      END                                                           08141
   ELSE mvdlfgrp( stid1, stid2, &vsptr, newstid, deltflag, FALSE,
   FALSE);                                                          08142
   RETURN;                                                          08143
   END.   %%
```

```
                                                                        08133
(cdelmar)   % GB: core NLS Delete Marker procedure %
PROCEDURE (tp1, tp2, fileno % => no value %);                           08836
   % Procedure description                                              08837
      FUNCTION                                                          08838
         Deletes the marker specified by the string between the text
         pointers from the file 'fileno'.                               08839
      ARGUMENTS                                                         08840
         tp1 - TEXT POINTER - start of marker name                      08841
         tp2 - TEXT POINTER - end of marker name                        09563
         fileno - INTEGER - NLS file number of the file                 09564
      RESULTS                                                           08842
         none                                                           08843
      NON-STANDARD CONTROL                                              08844
         Error if marker name has more than 5 characters                08845
      GLOBALS                                                           08846
         none                                                           08847
      EXAMPLE                                                           08848
         cdelmar($tp1, $tp2, stid.stfile)                               08849
      %                                                                 08850
   % Declarations %                                                     08851
      LOCAL mname, count, char;                                          0293
      LOCAL STRING astrng[25];                                           0294
      REF tp1, tp2;                                                      0295
   *astrng* _ tp1 tp2;                                                  08855
   astrng.L _ MIN(5, astrng.L);                                         08856
   CCPOS SF(*astrng*);                                                  08857
   count _ mname _ 0;                                                   08858
   LOOP                                                                 08859
      BEGIN                                                             08860
      CASE char _ READC OF                                              08861
         = ENDCHR: EXIT LOOP;                                           08862
         ENDCASE BUMP count;                                            08863
      CASE count OF                                                     08864
         = 1: mname.chr0 _ char;                                        08865
         = 2: mname.chr1 _ char;                                        08866
         = 3: mname.chr2 _ char;                                        08867
         = 4: mname.chr3 _ char;                                        08868
         = 5: mname.chr4 _ char;                                        08869
         ENDCASE err($"marker name too long");                          08870
      END;                                                              08871
   <AUXCOD, delmkn> (fileno, mname);                                    08872
   RETURN;                                                              08873
   END.   %%
```

```
                                                                    08854
(cdelmodfil)  % GB: core NLS Delete Modifications to File procedure %
PROCEDURE (fileno % => no value %);                                 07945
   % Procedure description                                          07946
      FUNCTION                                                      07947
         Deletes the Partial Copy assiciated with the NLS file
         'fileno', thereby deleting all the modifications to the file
         since it was last updated.  Then updates the file header to
         reflect the fact that the file is no longer locked (has no
         PC).                                                       07948
      ARGUMENTS                                                     07949
         fileno - INTEGER - NLS file number of the file             07950
      RESULTS                                                       07951
         none                                                       07952
      NON-STANDARD CONTROL                                          07953
         none                                                       07954
      GLOBALS                                                       07955
         none                                                       07956
      EXAMPLE                                                       07957
         cdelmodfil(stid.stfile)                                    07958
      %                                                             07959
   % Declarations %                                                 07960
   unlkfile(fileno, TRUE);                                          07961
   RETURN;                                                          07962
   END.   %%
```

```
                                                                    07963
(cdelsta)   % GB: core NLS Delete Statement procedure %
PROCEDURE (stid, filter, vsptr % => no value %);                    08093
   % Procedure description                                          08094
      FUNCTION                                                      08095
         Deletes the statement at 'stid'.  If 'filter' is TRUE,
         deletes the statement only if it passes the viewspecs in
         'vsptr'.                                                   09565
      ARGUMENTS                                                     08097
         stid - STID - statement to be deleted                      09566
         filter - BOOLEAN -                                         09574
            TRUE  => filter statements                              09575
            FALSE => delete all statements                          09576
         vsptr - VIEWSPECS -                                        09577
            viewspecs through which to filter statements, or 0.     09578
      RESULTS                                                       08099
         none                                                       08100
      NON-STANDARD CONTROL                                          08101
         Error if the statement has substructure (use 'cdelgro' in
         that case)                                                 08102
      GLOBALS                                                       08103
         none                                                       08104
      EXAMPLE                                                       08105
         cdelsta(stid, FALSE, 0)                                    08106
      %                                                             08107
   % Declarations %                                                08108
      REF vsptr;                                                     0323
   IF getsub(stid) # stid THEN err($"illegal delete");              08112
   cdelgro(stid, stid, filter, &vsptr);                             08113
   RETURN;                                                          08114
   END.   %%
```

```
                                                                    08111
(cdeltex)   % GB: core NLS Delete Text procedure %
PROCEDURE (bug1, bug2, includespace % => no value %);               08536
   % Procedure description                                          08537
      FUNCTION                                                      08538
         Deletes the text at 'bug1' through 'bug2'.  If includespace
         is TRUE, also deletes either a leading or a trailing space
         if there is one.                                           08539
      ARGUMENTS                                                     08540
         bug1 - TEXT POINTER - start of text to delete              08541
         bug2 - TEXT POINTER - end of text to delete                09579
         includespace - BOOLEAN -                                   09580
            TRUE => delete one extra space if there is one          09581
      RESULTS                                                       08542
         none                                                       08543
      NON-STANDARD CONTROL                                          08544
         none                                                       08545
      GLOBALS                                                       08546
         none                                                       08547
      EXAMPLE                                                       08548
         cdeltex($tp1, $tp2, FALSE)                                 08549
      %                                                             08550
   % Declarations %                                                 08551
      REF bug1, bug2;                                               0330
   IF includespace THEN incspc(&bug1, &bug2);                       08555
     %msntx();%                                                     08556
   cldtxt(&bug1, &bug2);                                            08557
   ST bug1 _ SF(bug1) bug1, bug2 SE(bug1);                          08558
     %msftx();%                                                     08559
   RETURN;                                                          08560
   END.   %%
```

```
                                                                    08554
(incspc)   % LB: sets text pointers to include spaces %
PROCEDURE (ptr1, ptr2 % => no value %);                             08874
   % Procedure description                                          08875
      FUNCTION                                                      08876
         none                                                       08877
      ARGUMENTS                                                     08878
         none                                                       08879
      RESULTS                                                       08880
         none                                                       08881
      NON-STANDARD CONTROL                                          08882
         none                                                       08883
      GLOBALS                                                       08884
         none                                                       08885
      EXAMPLE                                                       08886
         none                                                       08887
      %                                                             08888
   % Declarations %                                                08889
      REF ptr1, ptr2;                                                0339
   FIND ptr2 > (SP ^ptr2 / ptr1 < (SP ^ptr1 / TRUE));               08893
   RETURN;                                                          08894
   END.   %%
```

```
                                                                        08892
%expunge%                                                                0359
   (cexparcdir)   % UB: core NLS Expunge Archive Directory procedure %
   PROCEDURE %( => no value )%;                                         08895
      % Procedure description                                          08896
        FUNCTION                                                        08897
           Not implemented yet.                                         08898
        ARGUMENTS                                                       08899
           none                                                         08900
        RESULTS                                                         08901
           none                                                         08902
        NON-STANDARD CONTROL                                            08903
           none                                                         08904
        GLOBALS                                                         08905
           none                                                         08906
        EXAMPLE                                                         08907
           none                                                         08908
        %                                                               08909
      % Declarations %                                                  08910
        REF bug1, bug2;                                                  0362
      err( notyet );                                                    08914
      RETURN;                                                           08915
      END.   %%
```

```
                                                                    08913
(cexpcondir)   % GB: core NLS Expunge Connected Directory procedure %
PROCEDURE (exptype % => no value %);                                09321
   % Procedure description                                          09322
      FUNCTION                                                      09323
         Like 'cexpdir', except expunges only the connected
         directory, not the login directory.                        09324
      ARGUMENTS                                                     09325
         exptype - INTEGER - type of expunge:                       09326
            0   - all deleted files (for any reason)                09327
            4B7 - nonexistent files                                 09328
            2B7 - explicitly deleted files                          09329
            4B6 - scratch files                                     09330
            2B6 - temporary files                                   09331
      RESULTS                                                       09332
         none                                                       09333
      NON-STANDARD CONTROL                                          09334
         none                                                       09335
      GLOBALS                                                       09336
         none                                                       09337
      EXAMPLE                                                       09338
         cexpcondir(4B6)                                            09339
      %                                                             09340
   % expunge the connected directory %                              09346
      !gjinf();  %directory number in R2%                           09347
      deldfcall(exptype, r2);                                       09887
   % we're done, so return %                                       09365
      RETURN;                                                       09366
   END.   %%
```

```
                                                                    09367
(cexpdir)   % GB: core NLS Expunge Directory procedure %
PROCEDURE (exptype % => no value %);                                08074
   % Procedure description                                          08075
      FUNCTION                                                      08076
         Expunges (i.e. really throws away files) from both the
         connected and the login directories.  Works by first
         expunging the connected directory, then connecting to the
         login directory, then expunging that, then reconnecting to
         the previously connected directory.  The login directory is
         also expunged because NLS puts Partial Copies of all files
         on the login directory.                                    08077
         This has a potential pitfall: if connecting to a directory
         requires a password, this procedure will cause an error
         because it doesn't know the password.  It should only be
         used on directories that do not require a password when
         connecting to them.                                        09582
      ARGUMENTS                                                     08078
         exptype - INTEGER - type of expunge:                       08079
            0    - all deleted files (for any reason)               09313
            4B7 - nonexistent files                                 09314
            2B7 - explicitly deleted files                          09315
            4B6 - scratch files                                     09316
            2B6 - temporary files                                   09317
      RESULTS                                                       08080
         none                                                       08081
      NON-STANDARD CONTROL                                          08082
         Error if can't connect to login directory or reconnect to
         connected directory                                        08083
      GLOBALS                                                       08084
         chbmty - read only                                         08085
      EXAMPLE                                                       08086
         cexpdir(4B6)                                               08087
      %                                                             08088
   % Declarations %                                                 08089
      LOCAL                                                         07649
         logdirnum,  % login directory number %                     07650
         condirnum,  % connected directory number %                 07651
         bytptr;     % byte pointer for GTPSW and CNDIR jsies %     07652
   % get login and connected directory numbers %                    07658
      !gjinf();                                                     07659
      condirnum _ r2;                                               07661
   % connect to login directory (and expunge it) if needed %       07662
      IF logdirno # condirnum THEN                                  07663
         BEGIN                                                      07664
         % connect to login directory %                            07672
            IF NOT condc(logdirno, 0, 0) THEN                       07673
               err( $"can't connect to login directory" );          07674
         % now expunge the login directory %                       07675
            cexpcondir (exptype);                                   09318
         % now reconnect to the original connected directory %     07679
            IF NOT condc(condirnum, 0, $jpassw) THEN                07680
               err( $"can't reconnect to connected directory" );
                                                                    07681
         END;                                                       07682
   % expunge the connected directory %                             07683
```

```
    cexpcondir (exptype);                                   09320
% we're done, so return %                                  07687
    RETURN;                                                 07688
END.  %%
```

```
                                                                    08092
%insert%                                                             0412
   (cis)   % UB: old core NLS Insert Statement procedure %
   PROCEDURE (stid, string, levstr % => statement %);                08475
      % Procedure description                                       08476
         FUNCTION                                                   08477
            This is like 'cinssta', except that the arguments are
            strings instead of text pointers and integers.  It is kept
            around for historical reasons as the interface for old NLS
            calls.                                                  08478
         ARGUMENTS                                                  08497
            stid - STID -                                           08498
               statement after which to insert the new statement    08499
            string - STRING - string to make into a statement       08508
            levstr - STRING -                                       08509
               string containing d's and u's representing the level at
               which to insert the new statement                    09583
         RESULTS                                                    08506
            procedure-value - STID - new statement made             08507
         NON-STANDARD CONTROL                                       08483
            none                                                    08484
         GLOBALS                                                    08485
            none                                                    08486
         EXAMPLE                                                    08487
            cis(stid, $"Here is a new statement", $"uu")            08488
         %                                                          08489
      % Declarations %                                              08490
         LOCAL dir;                                                  0420
         LOCAL TEXT POINTER z1, z2;                                  0421
         REF string, levstr;                                         0422
      FIND SF(*string*) ^z1 SE(*string*) ^z2;                       08494
      stid _ levset(stid, &levstr : dir );                          08495
      RETURN( cinssta( stid, dir, $z1, $z2) );                      08496
      END.   %%
```

```
                                                                    08493
(cinssta)   % GB: core NLS Insert Statement procedure %
PROCEDURE (stid, rlevcnt, tp1, tp2 % => statement %);               08173
   % Procedure description                                         08174
      FUNCTION                                                      08175
         Makes a string into a statement and inserts it after 'stid'.
         'Rlevcnt' controls the level of the new statement.         08176
      ARGUMENTS                                                     08177
         stid - STID -                                              08178
            statement after which to insert the new statement       08199
         rlevcnt - INTEGER - level of the new statement             08200
            -1 => down one level from 'stid'                        08201
            0  => same level as 'stid'                              08202
            +n => up n levels from 'stid'                           08203
               Global variables levdown (-1), levsuc (0) and levup
               (+1) are recommended for this argument.              09584
         tp1 - TEXT POINTER - start of string                       08204
         tp2 - TEXT POINTER - end of string                         08205
      RESULTS                                                       08179
         procedure-value - STID - new statement made                08180
      NON-STANDARD CONTROL                                          08181
         Error if the two text pointers do not point to the same
         string or statement                                        08182
      GLOBALS                                                       08183
         none                                                       08184
      EXAMPLE                                                       08185
         cinssta(stid, levdown, $tp1, $tp2)                         08186
      %                                                             08187
   % Declarations %                                                08188
      LOCAL newstid; %new stid%                                      0429
      REF tp1, tp2;                                                  0430
   IF tp1 # tp2 THEN err($"Illegal Statement");                     08192
     %msnst();%                                                     08193
   newstid _ newrng(stid.stfile);                                   08194
   insgrp(stid, rlevcnt, newstid, newstid);                         08195
   ST newstid _ tp1 tp2;                                            08196
     %msfst();%                                                     08197
   RETURN(newstid);                                                 08198
   END.   %%
```

```
                                                                    08191
(cinstex)   % GB: core NLS Insert Text procedure %
PROCEDURE (tp1, tp2, tp3, insertspace % => no value %);             08510
   % Procedure description                                          08511
      FUNCTION                                                      08512
         Inserts the text at 'tp2' through 'tp3' to follow the
         character designated by 'tp1'.  If 'insertspace' is TRUE,
         then a space is inserted immediately after tp1 (i.e. before
         the inserted text).  This is actually the same as 'ccoptex'.
                                                                    08513
      ARGUMENTS                                                     08514
         tp1 - TEXT POINTER -                                       09590
            character after which to insert the new text            09591
         tp2 - TEXT POINTER - start of text to insert               09592
         tp3 - TEXT POINTER - end of text to insert                 09593
         insertspace - BOOLEAN - TRUE => insert a space first       09594
      RESULTS                                                       08516
         none                                                       08517
      NON-STANDARD CONTROL                                          08518
         none                                                       08519
      GLOBALS                                                       08520
         none                                                       08521
      EXAMPLE                                                       08522
         cinstex($tp1, $tp2, $tp3, TRUE)                            08523
      %                                                             08524
   % Declarations %                                                 08525
      REF tp1, tp2, tp3;                                             0440
     %msntx();%                                                     08529
   IF insertspace THEN                                              08530
      ST tp1 _ SF(tp1) tp1, SP,                                     08531
         $tp2 tp3,   % $ => don't move markers %                    09786
         tp1 SE(tp1)                                                09787
   ELSE                                                             08532
      ST tp1 _ SF(tp1) tp1,                                         08533
         $tp2 tp3,   % $ => don't move markers %                    09788
         tp1 SE(tp1);                                               09789
     %msftx();%                                                     08534
   RETURN;                                                          08535
   END.   %%
```

```
                                                                    08528
%load%                                                               0449
   (cloafil)   % GB: core NLS Load File procedure %
   PROCEDURE (flnam % => file number %);                            07919
      % Procedure description                                       07920
         FUNCTION                                                   07921
            Opens and loads a file into core, and returns the job file
            number (jfn) for the file.  Calls 'open' to do most of the
            work.                                                   07922
         ARGUMENTS                                                  07923
            flnam - STRING -                                        07943
               file name in TENEX format.  ".NLS" is assumed if an
               extension is not specified.  Connected directory is
               assumed if a directory is not specified.             07944
                  NOTE: The string is changed by this function to be the
                  full file name.  Therefore literal strings should not
                  be passed, as they will overflow; rather a STRING
                  VARIABLE large enough to hold the expanded name should
                  be passed.                                        09312
         RESULTS                                                    07925
            procedure-value - INTEGER -                             07926
               NLS file number (jfn) for the file                   09311
         NON-STANDARD CONTROL                                       07927
            Signals with 'ofilerr' if                               07928
               can't get a jfn for the file                         09598
               any subroutine generates an error                    09599
         GLOBALS                                                    07929
            none                                                    07930
         EXAMPLE                                                    07931
            LOCAL STRING filename[100];                             09309
            *filename* _ "<smith>foo.nls";                          07932
            stid.stfile _ cloafil($filename);                       09310
         %                                                          07933
      % Declarations %                                             07934
         LOCAL fileno, jfn;                                         02692
         REF flnam;                                                 02693
      % Get jfn (using long gtjfn) %                               02694
         IF NOT jfn _ lgetjfn (0, &flnam, $nlsext, gtjoif, $lit) THEN
                                                                    02695
            SIGNAL (ofilerr, $lit);                                 02696
      ON SIGNAL                                                     02697
         = errsig:                                                  02698
            SIGNAL (ofilerr, sysmsg);                               02699
         ELSE;                                                      02700
      %open the file if not now open--allocates a new file number if
      necessary--open pc if there is one%                           02701
         fileno _ open (jfn, &flnam);                               02702
      RETURN (fileno);                                              02703
      END.   %%
```

```
                                                                    07937
(cloamodfil)   % UWB: core NLS Load Modified File procedure %
PROCEDURE (filename % => statement %);                              08916
   % Procedure description                                          08917
      FUNCTION                                                      08918
         Note that this routine returns the STID of the origin
         statement, not just the file number.                       08919
         YOU MUST BE A "WHEEL" TO USE THIS PROCEDURE.               09603
      ARGUMENTS                                                     08920
         filename - STRING -                                        09600
            file name in TENEX format.  ".NLS" is assumed if an
            extension is not specified.  Connected directory is
            assumed if a directory is not specified.                09601
      RESULTS                                                       08922
         procedure-value - STID - origin statement of loaded file
                                                                    08923
      NON-STANDARD CONTROL                                          08924
         none                                                       08925
      GLOBALS                                                       08926
         none                                                       08927
      EXAMPLE                                                       08928
         none                                                       08929
      %                                                             08930
   % Declarations %                                                 08931
      REF filename;                                                  0458
   RETURN(openlock(0, &filename));                                  08935
   END.   %%
```

```
                                                                    08934
%logout%                                                             0461
   (clogout)   % GB: core NLS Logout procedure %
   PROCEDURE %( => no value )%;                                     08936
      % Procedure description                                      08937
         FUNCTION                                                   08938
            Logs out the user's job.                                08939
         ARGUMENTS                                                  08940
            none                                                    08941
         RESULTS                                                    08942
            none                                                    08943
         NON-STANDARD CONTROL                                       08944
            none                                                    08945
         GLOBALS                                                    08946
            <lots>                                                  08947
         EXAMPLE                                                    08948
            clogout()                                               08949
         %                                                          08950
      % Declarations %                                             08951
         LOCAL                                                      0464
            core,        %core page address%                        0465
            user,        %user job block address%                   0466
            corptr,      %address of block pointer array%           0467
            corgrp,      %address of allocation array%              0468
            corexp,      %address of express count aray%            0469
            ptr,         %temporary pointer%                        0470
            temp,        %temporary cell%                           0471
            n,           %counter%                                  0472
            jobno,       %job number%                               0473
            group,       %this jobs group no.%                      0474
            corjfn,      %jfn for data page%                        0475
            wleft,       %words left in userpgm buffer%            07289
            pn,          %file page number%                        07282
            cpn;         %file page number%                        07283
         REF user,corptr,corgrp,corexp,ptr;                         0476
      % if processing commands, reset recognition mode %           09780
         IF auxinput THEN                                           09781
            BEGIN                                                   09782
            recogmode _ auxmod;                                     09783
            recog2mode _ auxmd2;                                    09784
            END;                                                    09785
      % get core page file, open, map and lock it %                08955
         IF NOT SKIP !gtjfn(100001B6,chbmty+$"<system>group.core;1")
                                                                    08956
            THEN RETURN;                                            08957
         corjfn _ r1;                                               08958
         IF NOT SKIP !openf (corjfn,440000303000B) THEN             08959
            BEGIN                                                   08960
            !rljfn (corjfn);                                        08961
            RETURN;                                                 08962
            END;                                                    08963
         wleft_upgbend-upgffbuff+1; %make sure ther are 4 pages in
         userpgm buffer%                                            08964
         IF wleft<4000B THEN                                        08965
            BEGIN                                                   08966
            gpbsz(upgbsz + (4000B-wleft)/512 + 1);                  08967
```

```
      END;                                                          08968
   core _ (upgffbuff+777B) .A 777000B;                              08969
   cpn _ core/1000B;                                                08970
   r1.LH _ corjfn;                                                  08971
   r1.RH _ 0;                                                       08972
   FOR pn _ 0 UP UNTIL =3 DO                                        08973
      BEGIN                                                         08974
      !pmap(r1, 4B11 + cpn + pn, 140000B6);                         08975
      BUMP r1                                                       08976
      END;                                                          08977
   IF NOT lockpage(core) THEN RETURN; %lock the data page%          08978
   !gjinf; %get this guys job number%                               08979
   jobno _ r3;                                                      08980
% setup  pointers %                                                 08981
   &corptr _ ptradr+core;  %point to block ptr array%               08982
   &corgrp _ grpadr+core;  %point to alloc/logged array%            08983
   &corexp _ expadr+core;  %point to express count array%           08984
   %point to this jobs block%                                       08985
   &user _ (jobno * 10B) + corblk + core;                           08986
   group _ user.usrgrp;                                             08987
% now delete this job from the data page %                         08988
   IF (user.usrflgs .A usrlgd) THEN                                 08989
      BEGIN                                                         08990
      CASE 0 OF                                                     08991
         = user.usrlnk:  %check if the chain pointer agrees this
         is only guy%                                               08992
            BEGIN                                                   08993
            temp _ &user - core; % compute the displacement%        08994
            temp.LH _ temp.RH;                                      08995
            IF temp # corptr[group] THEN                            08996
               BEGIN                                                08997
               [core] _ -1;        %unlock the data page%           08998
               RETURN;                                              08999
               END;                                                 09000
            corptr[group] _ 0;     %say none in chain%              09001
            END;                                                    09002
         = user.usrlnk.RH:  %no forward link, this is the end%
                                                                    09003
            BEGIN                                                   09004
            corptr[group].LH _ user.usrlnk.LH;                      09005
            &ptr _ core + user.usrlnk.LH; %compute back addr%
                                                                    09006
            ptr.usrlnk.RH _ 0;  %make back new end%                 09007
            END;                                                    09008
         = user.usrlnk.LH:  %no back link, this is the head%        09009
            BEGIN                                                   09010
            corptr[group].RH _ user.usrlnk.RH;                      09011
            &ptr _ core + user.usrlnk.RH; %compute forward addr%
                                                                    09012
            ptr.usrlnk.LH _ 0;  %make forward new head%             09013
            END;                                                    09014
         ENDCASE  %in the middle of a chain unhook him%             09015
            BEGIN                                                   09016
            &ptr _ core+user.usrlnk.RH;                             09017
            ptr.usrlnk.LH _ user.usrlnk.LH;                         09018
            &ptr _ core+user.usrlnk.LH;                             09019
```

```
            ptr.usrlnk.RH _ user.usrlnk.RH;                          09020
            END;                                                     09021
         IF (user.usrflgs .A usrexp) THEN                            09022
            %decr express counts%                                    09023
            BEGIN                                                    09024
            BUMP DOWN corexp[group];                                 09025
            BUMP DOWN [core+corlge];                                 09026
            %update total runtime for express guys%                  09027
            !runtm(-5); %get total runtime for this job%             09028
            r1 _ r1 / r2;  %convert to seconds%                      09029
            [core+corexr] _ [core+corexr] + r1;                      09030
            END                                                      09031
         ELSE                                                        09032
            %decr regular counts%                                    09033
            BEGIN                                                    09034
            BUMP DOWN corgrp[group];                                 09035
            BUMP DOWN [core+corlgd];                                 09036
            END;                                                     09037
         % now clear user block %                                    09038
         FOR n _ 0 UP UNTIL = 7 DO [&user + n] _ 0;                  09039
         END;                                                        09040
      [core] _ -1;   %unlock the data page%                          09041
   RETURN;                                                           09042
   END.   %%
```

```
                                                                    08954
%mark%                                                               0556
   (cmarcha)   % GB: core NLS Mark Character procedure %
   PROCEDURE (bug, tp1, tp2 % => no value %);                       09043
      % Procedure description                                       09044
         FUNCTION                                                   09045
            Inserts a marker pointing to the character at "bug".  If the
            marker already exists, it is changed to point to the new
            character.                                              09046
         ARGUMENTS                                                  09047
            bug - TEXT POINTER - character to mark                  09048
            tp1 - TEXT POINTER - start of marker name               09604
            tp2 - TEXT POINTER - end of marker name                 09605
         RESULTS                                                    09049
            none                                                    09050
         NON-STANDARD CONTROL                                       09051
            none                                                    09052
         GLOBALS                                                    09053
            origin, rngtype - read only                             09054
         EXAMPLE                                                    09055
            cmarcha($tp1, $tp2, $tp3)                               09056
         %                                                          09057
      % Declarations %                                              09058
         LOCAL mname, count, length, stid, aring;                   06735
         LOCAL STRING astrng[25];                                   06736
         REF bug, tp1, tp2, aring;                                  06737
      % make sure we have a locked file %                           09062
         stid _ origin;                                             09063
         stid.stfile _ bug.stfile;                                  09064
         lodent( stid, rngtyp : &aring);                            09065
         aring.rsub _ aring.rsub;                                   09066
      *astrng* _ tp1 tp2;                                           09067
      count _ mname _ 0;                                            09068
      length _ MIN (5, astrng.L);                                   09069
      UNTIL (count _ count+1) > length DO                           09070
         CASE count OF                                              09071
            = 1: mname.chr0 _ *astrng*[count];                      09072
            = 2: mname.chr1 _ *astrng*[count];                      09073
            = 3: mname.chr2 _ *astrng*[count];                      09074
            = 4: mname.chr3 _ *astrng*[count];                      09075
            = 5: mname.chr4 _ *astrng*[count];                      09076
            ENDCASE;                                                09077
      <AUXCOD, insmkr> (&bug, mname);                               09078
      RETURN;                                                       09079
      END.   %%
```

```
                                                                    09061
%merge%                                                              0577
   % COMMENTED OUT TO SAVE SPACE IN SYSTEM -- percents doubled       09791
   (cmerge)   %% LB: core NLS Merge primitive %%
   PROCEDURE (outlst, inlst, kpr, percnt, copyf %% => no value %%);
                                                                    09080
      %% Procedure description                                     09081
         FUNCTION                                                   09082
            none                                                    09083
         ARGUMENTS                                                  09084
            none                                                    09085
         RESULTS                                                    09086
            none                                                    09087
         NON-STANDARD CONTROL                                       09088
            none                                                    09089
         GLOBALS                                                    09090
            none                                                    09091
         EXAMPLE                                                    09092
            none                                                    09093
         %%                                                         09094
      %% Declarations %%                                           09095
      <COLSRT, merge> (outlst,inlst,kpr,percnt,copyf);              09099
      RETURN;                                                       09097
      END.   %%%%
```

```
                                                                    09098
(cmergro)   %% UB: core NLS Merge Group procedure %%
PROCEDURE (intobug1, intobug2, frombug1, frombug2 %% => no value %%);
                                                                    08450
   %% Procedure description                                        08451
      FUNCTION                                                      08452
         none                                                       08453
      ARGUMENTS                                                     08454
         none                                                       08455
      RESULTS                                                       08456
         none                                                       08457
      NON-STANDARD CONTROL                                          08458
         none                                                       08459
      GLOBALS                                                       08460
         none                                                       08461
      EXAMPLE                                                       08462
         none                                                       08463
      %%                                                            08464
   %% Declarations %%                                              08465
      REF intobug1,intobug2,frombug1,frombug2;                      08474
   <CULSRT, merggrp> (&intobug1,&intobug2,&frombug1,&frombug2);     08469
   RETURN;                                                          08467
   END.   %%%%
```

```
                                                                    08468
(cupdate)   %% LB: core NLS Update primitive %%
PROCEDURE (outlst, inlst, kpr, udpr, percnt, initf %% => no value
%%);                                                                09100
   %% Procedure description                                        09101
      FUNCTION                                                      09102
         none                                                       09103
      ARGUMENTS                                                     09104
         none                                                       09105
      RESULTS                                                       09106
         none                                                       09107
      NON-STANDARD CONTROL                                          09108
         none                                                       09109
      GLOBALS                                                       09110
         none                                                       09111
      EXAMPLE                                                       09112
         none                                                       09113
      %%                                                            09114
   %% Declarations %%                                              09115
   <COLSRT, update> (outlst,inlst,kpr,udpr,percnt,initf);           09119
   RETURN;                                                          09117
   END.   %%%%
```

```
                                                                    09118
   %                                                                09790
%move%                                                               0591
   (cmovfil)   % GB: core NLS Move File procedure %
   PROCEDURE (rhost1, file1, rhost2, file2, astr % => no value %);  07986
      % Procedure description                                       07987
         FUNCTION                                                   07988
            Moves the NLS file(s) specified by 'file1' on host 'rhost1'
            to file(s) 'file2' on host 'rhost2'.  Any partial copies are
            also moved.  Asterisks (*) in the source file specification
            (file1) must be matched by asterisks in the destination file
            specification (file2).  Messages describing the results are
            put in 'astr'.  Note: this moves the source file's origin
            statement along with its other statements.              09628
         ARGUMENTS                                                  07990
            rhost1 - INTEGER -                                      09618
               number of the host on which the files reside.  Only the
               local host ('lhostn') is currently implemented.      09619
            file1 - STRING -                                        09613
               file group name string in TENEX format.  Connected
               directory is assumed if a directory is not specified.
               Examples:                                            09614
                  "*.*"                                             09615
                  "<SMITH>*.PC"                                     09616
                  "<SMITH>FOO.NLS;3"                                09617
            rhost2 - INTEGER -                                      09611
               number of the host to which the files are to be moved.
               Only the local host ('lhostn') is currently implemented.
                                                                    09612
            file2 - STRING -                                        09606
               file group name string in TENEX format.  Connected
               directory is assumed if a directory is not specified.
               Examples:                                            09607
                  "*.*"                                             09608
                  "<SMITH>*.PC"                                     09609
                  "<SMITH>FOO.NLS;3"                                09610
               Note: the *'s here must match one-for-one with the *'s in
               file1.                                               09620
            astr - STRING - string to receive the file names        09621
         RESULTS                                                    07992
            none                                                    07993
         NON-STANDARD CONTROL                                       07994
            Error if                                                07995
               <CTRL-F> or <ALTMODE> is used in 'file2'             09622
               *'s don't match                                      09623
               input files don't exist                              09624
               output files are deleted but not expunged            09625
               other file-moving problems, e.g. protection violations
                                                                    09626
         GLOBALS                                                    07996
            <lots>                                                  07997
         EXAMPLE                                                    07998
            cmovfil(lhostn, $"*.NLS;*", lhostn, $"<SMITH>*.NLS;*",
            $string)                                                07999
         %                                                          08000
      % Declarations %                                             08001
```

```
    LOCAL                                                          06223
       jfn1,        % main jfn %                                    06224
       jfn2,        % jfn for destination file %                    06225
       jfnfl1,      % left half flags for a group jfn %             06226
       jfnfl2,      % left half flags for destination group jfn %
                                                                   06227
       pcjfn1,      % jfn of partial copy %                         06228
       pcjfn2,      % jfn of partial copy %                         06229
       flag1,       % bits indicating * typed for file name fields %
                                                                   06230
       flag2;       % * typed for destination file name fields %
                                                                   06231
    LOCAL STRING                                                   06233
       uf1[200],    % complete name as entered by user %            06234
       jfnnm1[200],% complete name from jfns %                     06235
       uf2[200],    % complete destination name as entered by user %
                                                                   06236
       udir2[40],   % user input directory destination name %      06237
       ufil2[40],   % user input file destination name %           06238
       uext2[40],   % user input extension destination name %      06239
       uver2[40],   % user input destination version number %      06240
       ulft2[40],   % user destination input beyond version number %
                                                                   06241
       jfnnm2[200],         % complete destination name from jfns %
                                                                   06242
       errstring[200],      % eror message string %                 06243
       tstring[200];        % temporary string %                    06244
    REF file1, file2, astr;                                        06245
% initial variables %                                             06247
    jfn1 _ jfnfl1 _ pcjfn1 _ flag1 _ 0;                            06248
    jfn2 _ jfnfl2 _ pcjfn2 _ flag2 _ 0;                            06249
% find out if remote or local (disk) file %                       06250
    CASE rhost1 OF                                                 06251
       = lhostn:  NULL;                                            06252
       ENDCASE err( $"remote file manipulations not implemented
       yet" );                                                     06253
    CASE rhost2 OF                                                 06254
       = lhostn:  NULL;                                            06255
       ENDCASE err( $"remote file manipulations not implemented
       yet" );                                                     06256
% now parse user source input string %                            06257
    parseinput( &file1, $udir2, $ufil2, $uext2, $uver2, $ulft2,
    $uf1, $flag1);                                                 06258
% parse user destination input string %                           06259
    parseinput( &file2, $udir2, $ufil2, $uext2, $uver2, $ulft2,
    $uf2, $flag2);                                                 06260
    % illegal to terminate destination fields with ^F or alt %
                                                                   06261
       CASE *udir2*[udir2.L] OF                                    06262
          = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
                                                                   06263
          ENDCASE;                                                 06264
       CASE *ufil2*[ufil2.L] OF                                    06265
          = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
                                                                   06266
          ENDCASE;                                                 06267
```

```
        CASE *uext2*[uext2.L] OF                                        06268
           = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
                                                                        06269
           ENDCASE;                                                     06270
        CASE *uver2*[uver2.L] OF                                        06271
           = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
                                                                        06272
           ENDCASE;                                                     06273
        CASE *ulft2*[ulft2.L] OF                                        06274
           = '<^F>, = '<ESC>: err($"Illegal use of ^F or ALTMODE");
                                                                        06275
           ENDCASE;                                                     06276
    % check for matching astericks %                                    06277
      IF flag1.dstar AND NOT flag2.dstar THEN err($"Illegal use of
      *");                                                              06278
      IF flag1.fstar AND NOT flag2.fstar THEN err($"Illegal use of
      *");                                                              06279
      IF flag1.estar AND NOT flag2.estar THEN err($"Illegal use of
      *");                                                              06280
      IF flag1.vstar AND NOT flag2.vstar THEN err($"Illegal use of
      *");                                                              06281
    % get main source jfn (old file only, group jfn) %                  06282
      IF NOT (jfn1 _ sgtjfn( gtjoif .V gtjstr + 0, $uf1, $errstring :
      jfnfl1 ) )THEN err( $errstring );                                 06283
    % now find out if we support this type of file %                    06284
      IF NOT chkdev( jfn1 ) THEN err( $"only disk files supported" );
                                                                        06285
    % now loop to get all files in the group %                          06286
      LOOP                                                              06287
         BEGIN                                                          06288
         % check to make sure this not a PC with * for extension name
         %                                                              06289
            IF NOT chkpcs( jfn1, flag1) THEN GOTO gtmnxtjfn;            06290
         % get actual file name into string for use later (maybe) %
                                                                        06291
            r3 _ 01111086     %  directory file ext ver %              06292
               + (IF jfnfl1.lhjb9  THEN 1B6 ELSE 0) % ;Protection %
                                                                        06293
               + (IF jfnfl1.lhjb10 THEN 1B5 ELSE 0) % ;Account %
                                                                        06294
               + (IF jfnfl1.lhjb11 THEN 4B4 ELSE 0) % ;T %              06295
               + 1;           % with proper file punctuation %        06296
            jfnflink( jfn1, $jfnnm1, r3);                               06297
         % get source PC jfn & find if this user can access this PC %
                                                                        06298
            IF NOT getpcjfn( jfn1, $pcjfn1, $errstring, FALSE) THEN
                                                                        06299
               BEGIN % this user can't access this file %               06300
               *tstring* _ *jfnnm1*, " not moved because [",
               *errstring*, ']';                                        06301
               IF flag1 THEN                                            06302
                  BEGIN                                                 06303
                  *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
                                                                        06304
                  GOTO gtmnxtjfn;                                       06305
                  END                                                   06306
```

```
        ELSE                                                    06307
           BEGIN                                                06308
           IF NOT SKIP !rljfn( jfn1) THEN NULL;                 06309
           err( $tstring );                                     06310
           END;                                                 06311
        END;                                                    06312
     % get destination jfns %                                   06313
      IF NOT jfn2 _                                             06314
        gtdesjfn( jfn1, pcjfn1, $udir2, $ufil2, $uext2,
        $uver2, $ulft2, flag2, $errstring : pcjfn2) THEN        06315
           BEGIN      % can't move file (or partial copy) %
                                                                06316
           *tstring* _ *jfnnm1*, " not moved because ",
           *errstring*;                                         06317
           IF flag1 THEN                                        06318
              BEGIN                                             06319
              *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
                                                                06320
              GOTO gtmnxtjfn;                                   06321
              END                                               06322
           ELSE                                                 06323
              BEGIN                                             06324
              IF NOT SKIP !rljfn( jfn1) THEN NULL;              06325
              IF NOT SKIP !rljfn( pcjfn1) THEN NULL;            06326
              err( $tstring );                                  06327
              END;                                              06328
           END;                                                 06329
     % get destination file name into string for use later %
                                                                06330
      r3 _ 011110B6    %  directory file ext ver %             06331
        + (IF jfnfl1.lhjb9  THEN 1B6 ELSE 0) % ;Protection %
                                                                06332
        + (IF jfnfl1.lhjb10 THEN 1B5 ELSE 0) % ;Account %
                                                                06333
        + (IF jfnfl1.lhjb11 THEN 4B4 ELSE 0) % ;T %             06334
        + 1;          % with proper file punctuation %         06335
      jfnflink( jfn2, $jfnnm2, r3);                             06336
     % now move the file (and its partial copy) %              06337
      IF NOT movflandpc(jfn1, pcjfn1, jfn2, pcjfn2, $errstring)
      THEN                                                      06338
        BEGIN % can't move file (or partial copy) %             06339
        *tstring* _ *jfnnm1*, " not moved because ",
        *errstring*;                                            06340
        IF flag1 THEN                                           06341
           BEGIN                                                06342
           *astr* _ *astr*, "   ***   ", *tstring*, CR, LF;
                                                                06343
           GOTO gtmnxtjfn;                                      06344
           END                                                  06345
        ELSE                                                    06346
           BEGIN                                                06347
           IF NOT SKIP !rljfn( jfn1) THEN NULL;                 06348
           IF NOT SKIP !rljfn( pcjfn1) THEN NULL;               06349
           err( $tstring );                                     06350
           END;                                                 06351
        END;                                                    06352
```

```
        % update the file return ring %                                  06353
          uplsfnm( $jfnnm1, $jfnnm2);                                    06354
        % now tell the user we have moved this file %                     06355
          *astr* _ *astr*, "   ", *jfnnm1*;                              06356
          IF pcjfn1 THEN *astr* _ *astr*, " [and PC]";                   06357
          *astr* _ *astr*, " to ", *jfnnm2*;                             06358
          IF pcjfn2 THEN *astr* _ *astr*, " [and PC]";                   06359
          *astr* _ *astr*, CR, LF;                                       06360
        % now get rid of the jfn for the partial %                       06361
          IF NOT SKIP !rljfn( pcjfn1 ) THEN NULL;                        06362
        % now get the next file in the group %                           06363
          (gtmnxtjfn):                                                   06364
          r1.LH _ jfnfl1;                                                06365
          r1.RH _ jfn1;                                                  06366
          IF NOT SKIP !gnjfn( r1 ) THEN EXIT LOOP;                       06367
        END;                                                             06368
% now get rid of any lingering jfns %                                    06369
    IF NOT SKIP !rljfn( jfn1 ) THEN NULL;                                06370
    IF NOT SKIP !rljfn( pcjfn1 ) THEN NULL;                              06371
% all done now, so return %                                             06372
   RETURN;                                                               06373
END.   %%
```

```
                                                                    08004
(cmovgro)   % GB: core NLS Move Group procedure %
PROCEDURE (tostid, rlevcnt, fromstid1, fromstid2, filter, vsptr % =>
statement, statement %);                                            08358
   % Procedure description                                         08359
      FUNCTION                                                      08360
         Moves the group at 'fromstid1' through 'fromstid2' to follow
         'tostid' at the relative level 'rlevcnt'.  If 'filter' is
         TRUE, moves only those statements which pass the viewspecs
         in 'vsptr'.                                              09630
      ARGUMENTS                                                     08362
         tostid - STID -                                          09631
            statement after which to insert the new group         09632
         rlevcnt - INTEGER - level of the new group               09633
            -1 => down one level from 'tostid'                    09634
            0  => same level as 'tostid'                          09635
            +n => up n levels from 'tostid'                       09636
               Global variables levdown (-1), levsuc (0) and levup
               (+1) are recommended for this argument.            09637
         fromstid1 - STID - head of group to move                 09638
         fromstid2 - STID - tail of group to move                 09639
         filter - BOOLEAN -                                       09640
            TRUE  => filter statements                            09641
            FALSE => move all statements                          09642
         vsptr - VIEWSPECS -                                      09643
            viewspecs through which to filter statements, or 0.   09644
      RESULTS                                                       08398
         procedure-value - STID - head of group moved             08399
         2nd-result - STID - tail of group moved                  09645
      NON-STANDARD CONTROL                                          08366
         Error if                                                 08367
            destination of the group is inside the group itself   09646
            not a legal group                                     09647
      GLOBALS                                                       08368
         moveflag, endfil - read only                             08369
      EXAMPLE                                                       08370
         head _                                                   08371
            cmovgro(stid1, levdown, stid2, stid3, FALSE, 0 : tail)
                                                                    09650
      %                                                             08372
   % Declarations %                                               08373
      LOCAL newhead, newtail, work;                                 06432
      REF vsptr;                                                    06433
   fromstid1 _ grptst(fromstid1, fromstid2 : fromstid2);            09670
   movtst(tostid, fromstid1, fromstid2);                            09671
   IF tostid.stfile = fromstid1.stfile THEN                         09672
      BEGIN %intrafile move%                                        09673
      IF NOT filter THEN                                            09674
         BEGIN                                                      09675
         IF NOT remgrp(fromstid1, fromstid2) THEN err($"illegal
         move");                                                  09676
         insgrp(tostid, rlevcnt, fromstid1, fromstid2);           09677
         END                                                        09678
      ELSE                                                          09679
         mvdlfgrp( fromstid1, fromstid2, &vsptr,                  09680
            IF (work _ getnxt(getend(fromstid2))) # endfil THEN work
```

```
            ELSE getbck(fromstid1), moveflag, tostid, rlevcnt);  09681
      newhead _ fromstid1;  newtail _ fromstid2;                 09682
      END                                                        09683
   ELSE                                                          09684
      BEGIN %cross file move%                                    09685
      newhead _                                                  09686
         ccopgro(tostid, rlevcnt, fromstid1, fromstid2, filter,
         &vsptr: newtail);                                       09687
      cdelgro(fromstid1, fromstid2, filter, &vsptr);             09688
      END;                                                       09689
   RETURN(newhead, newtail);                                     09690
   END.
                                                                 09691
(movtst)   % LB: tests a group for movability %
PROCEDURE (stid, grp1, grp2 % => no value %);                    08400
   % Procedure description                                      08401
      FUNCTION                                                   08402
         Given an stid as the first argument, this routine makes sure
         that the stid is not in the group bounded by the second and
         third arguments passed it.  If it is, it does an err.   08448
         First it sees whether STID is a suc of GRP2 on the same
         level.  If not, it then sees whether STID's sucs (and ups)
         are GRP1 or GRP2.                                       08449
      ARGUMENTS                                                  08404
         none                                                    08405
      RESULTS                                                    08406
         none                                                    08407
      NON-STANDARD CONTROL                                       08408
         Generates errors if the move would be illegal.          08409
      GLOBALS                                                    08410
         none                                                    08411
      EXAMPLE                                                    08412
         none                                                    08413
      %                                                          08414
   % Declarations %                                              08415
      LOCAL tmpsid; %working stid%                                0624
   IF grp2.stpsid = origin THEN err($"illegal move");            08419
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");      08420
   IF stid.stfile # grp1.stfile                                  08421
      OR stid.stpsid = origin THEN RETURN;                       08422
   IF stid = grp1 OR stid = grp2 THEN err($"illegal move");      08423
   tmpsid _ grp2;                                                08424
   LOOP %is stid in plex after grp2%                             08425
      BEGIN                                                      08426
      IF getftl(tmpsid) THEN                                     08427
         BEGIN                                                   08428
         tmpsid _ getsuc(tmpsid);                                08429
         EXIT;                                                   08430
         END;                                                    08431
      tmpsid _ getsuc(tmpsid);                                   08432
      IF stid = tmpsid THEN RETURN; %psid in plex after grp2%    08433
      END;                                                       08434
   LOOP %see if grp1, grp2 in superstructure of psid%            08435
      BEGIN                                                      08436
      IF stid = grp1 THEN err($"illegal move");                  08437
      IF stid.stpsid = origin OR stid = tmpsid THEN RETURN;      08438
```

```
    LOOP                                                              08439
        BEGIN                                                         08440
        IF stid = grp2 THEN err($"illegal move");                     08441
        IF getftl(stid) THEN EXIT;                                    08442
        stid _ getsuc(stid);                                          08443
        IF stid = grp1 THEN RETURN;                                   08444
        END;                                                          08445
     stid _ getsuc(stid);                                             08446
     END;                                                             08447
END.    %%
```

```
                                                                    08418
(cmovsta)   % GB: core NLS Move Statement procedure %
PROCEDURE (tostid, rlevcnt, fromstid, filter, vsptr % => statement
%);                                                                 08335
   % Procedure description                                         08336
      FUNCTION                                                     08337
         Moves the statement at 'fromstid' to follow 'tostid' at the
         relative level 'rlevcnt'.  If 'filter' is TRUE, moves the
         statement only if it passes the viewspecs in 'vsptr'.  Will
         not move a statement with substructure; use 'cmovgro' in
         that case.                                                08338
      ARGUMENTS                                                    08339
         tostid - STID -                                           09651
            statement after which to insert the new statement      09652
         rlevcnt - INTEGER - level of the new statement            09653
            -1 => down one level from 'tostid'                     09654
            0  => same level as 'tostid'                           09655
            +n => up n levels from 'tostid'                        09656
               Global variables levdown (-1), levsuc (0) and levup
               (+1) are recommended for this argument.             09657
         fromstid - STID - statement to move                       09658
         filter - BOOLEAN -                                        09660
            TRUE  => filter statements                             09661
            FALSE => move all statements                           09662
         vsptr - VIEWSPECS -                                       09663
            viewspecs through which to filter statements, or 0.    09664
      RESULTS                                                      08341
         procedure-value - STID - statement moved                  08342
      NON-STANDARD CONTROL                                         08343
         Error if the statement has substructure                   08344
      GLOBALS                                                      08345
         none                                                      08346
      EXAMPLE                                                      08347
         cmovsta(stid1, levdown, stid2, FALSE, 0)                  08348
      %                                                            08349
   % Declarations %                                               08350
      REF vsptr;                                                    0657
   IF getsub(fromstid) # fromstid THEN                             08354
      err($"illegal move");                                        08355
   tostid _ cmovgro(tostid, rlevcnt, fromstid, fromstid, filter,
   &vsptr);                                                        08356
   RETURN(tostid);                                                 08357
   END.   %%
```

```
                                                                    08353
(cmovtex)   % GB: core NLS Move Text procedure %
PROCEDURE (totptr, fromtptr1, fromtptr2, insertspace % => no value
%);                                                                 08604
   % Procedure description                                         08605
      FUNCTION                                                     08606
         Moves the text at 'fromtptr1' through 'fromtptr2' to follow
         the character designated by 'totptr'.  If 'includespace' is
         TRUE, a space is inserted at the front of the new text
         location and deleted (if there is one) from the right or
         left end of the old text location.                        08607
      ARGUMENTS                                                    08608
         totptr - TEXT POINTER -                                   09665
            character after which to insert the new text           09666
         fromtptr1 - TEXT POINTER - start of text to move          09667
         fromtptr2 - TEXT POINTER - end of text to move            09668
         insertspace - BOOLEAN - TRUE => insert a space first      09669
      RESULTS                                                      08610
         none                                                      08611
      NON-STANDARD CONTROL                                         08612
         none                                                      08613
      GLOBALS                                                      08614
         none                                                      08615
      EXAMPLE                                                      08616
         cmovtex($tp1, $tp2, $tp3, FALSE)                          08617
      %                                                            08618
   % Declarations %                                               08619
      LOCAL TEXT POINTER del1, del2;                               0665
      LOCAL STRING locstr[5];                                      0666
      REF totptr, fromtptr1, fromtptr2;                            0667
     %msntx();%                                                    0668
   del1 _ fromtptr1;  del1[1] _ fromtptr1[1];                      0669
   del2 _ fromtptr2;  del2[1] _ fromtptr2[1];                      0670
   IF insertspace THEN                                             0671
      BEGIN                                                        0672
      *locstr* _ SP;                                               0673
      incspc($del1, $del2);                                        0674
      END                                                          0675
   ELSE *locstr* _ NULL;                                           0676
   IF POS SF(totptr) = SF(fromtptr1) THEN                          0677
      IF POS totptr < fromtptr1 THEN                               0678
         ST totptr _ SF(totptr) totptr, *locstr*, fromtptr1
         fromtptr2,                                                0679
            totptr del1, del2 SE(totptr)                           0680
      ELSE                                                         0681
         ST totptr _ SF(totptr) del1, del2 totptr,                 0682
            *locstr*, fromtptr1 fromtptr2, totptr SE(totptr)       0683
   ELSE                                                            0684
      BEGIN                                                        0685
      ST totptr _ SF(totptr) totptr,                               0686
         *locstr*, fromtptr1 fromtptr2, totptr SE(totptr);         0687
      ST fromtptr1 _ SF(fromtptr1) del1, del2 SE(fromtptr1)        0688
      END;                                                         0689
     %msftx();%                                                    0690
   RETURN;                                                         0691
   END.   %%
```

```
                                                                        08622
%output%                                                                 0693
   (coutassfil)   % UB: core NLS Output to Assembler File procedure %
   PROCEDURE (ofilnam, da, forceup % => no value %);                    09120
      % Procedure description                                          09121
         FUNCTION                                                      09122
            none                                                       09123
         ARGUMENTS                                                     09124
            none                                                       09125
         RESULTS                                                       09126
            none                                                       09127
         NON-STANDARD CONTROL                                          09128
            none                                                       09129
         GLOBALS                                                       09130
            none                                                       09131
         EXAMPLE                                                       09132
            none                                                       09133
         %                                                             09134
      % Declarations %                                                 09135
         REF ofilnam, da;                                             0696
      <SEQFIL, outasm> (&ofilnam, &da, forceup);                       09139
      RETURN;                                                          09140
      END.   %%
```

```
                                                                    09138
(coutjouqui)   % UB: core NLS Output Journal Quickprint procedure %
PROCEDURE (ofilnam, da % => no value %);                            09160
   % Procedure description                                          09161
      FUNCTION                                                      09162
         none                                                       09163
      ARGUMENTS                                                     09164
         none                                                       09165
      RESULTS                                                       09166
         none                                                       09167
      NON-STANDARD CONTROL                                          09168
         none                                                       09169
      GLOBALS                                                       09170
         none                                                       09171
      EXAMPLE                                                       09172
         none                                                       09173
      %                                                             09174
   % Declarations %                                                09175
      REF ofilnam, da;                                               0708
   <SEQFIL, outjm> (&ofilnam, &da);                                 09176
   RETURN;                                                          09177
   END.   %%
```

```
                                                                    09178
(coutqui)   % UB: core NLS Output Quickprint procedure %
PROCEDURE (ofilnam, da % => no value %);                            09179
   % Procedure description                                          09180
      FUNCTION                                                      09181
         none                                                       09182
      ARGUMENTS                                                     09183
         none                                                       09184
      RESULTS                                                       09185
         none                                                       09186
      NON-STANDARD CONTROL                                          09187
         none                                                       09188
      GLOBALS                                                       09189
         none                                                       09190
      EXAMPLE                                                       09191
         none                                                       09192
      %                                                             09193
   % Declarations %                                                09194
      REF ofilnam, da;                                               0720
   <SEQFIL, outqp> (&ofilnam, &da);                                 09195
   RETURN;                                                          09196
   END.   %%
```

```
                                                                        09197
(coutseqfil)   % UB: core NLS Output Sequential File procedure %
PROCEDURE (ofilnam, da, forceup % => no value %);                       09141
   % Procedure description                                              09142
      FUNCTION                                                          09143
         none                                                           09144
      ARGUMENTS                                                         09145
         none                                                           09146
      RESULTS                                                           09147
         none                                                           09148
      NON-STANDARD CONTROL                                              09149
         none                                                           09150
      GLOBALS                                                           09151
         none                                                           09152
      EXAMPLE                                                           09153
         none                                                           09154
      %                                                                 09155
   % Declarations %                                                     09156
      REF ofilnam, da;                                                   0732
   <SEQFIL, outseq> (&ofilnam, &da, forceup);                           09157
   RETURN;                                                              09158
   END.   %%
```

```
                                                                    09159
(coutproc)   % UB: core NLS invoke Output Processor %
PROCEDURE (ofilnam, da, device, opflags % => no value %);           09794
   % Procedure description                                          09795
      FUNCTION                                                      09796
         none                                                       09797
      ARGUMENTS                                                     09798
         ofilnam - STRING - name of the output file                 09799
         da - DISPLAY AREA - display area for viewspecs, etc.       09800
         device - INTEGER - output device                           09801
         opflags - INTEGER - device dependant flags for OP          09802
      RESULTS                                                       09803
         none                                                       09804
      NON-STANDARD CONTROL                                          09805
         none                                                       09806
      GLOBALS                                                       09807
         none                                                       09808
      EXAMPLE                                                       09809
         none                                                       09810
      %                                                             09811
   % Declarations %                                                 09812
      LOCAL jfn, errors, prcjfn , opentype;                         09813
      LOCAL gproc;  %graphics procedure called by U.P for COM %     09814
      LOCAL TEXT POINTER tp;                                        09815
      LOCAL STRING gname[20];  %name of graphics user program%      09816
      REF ofilnam, da;                                              09817
   % open the output file %                                        09818
      IF (NOT oqapfg) OR ( FIND SF(*ofilnam*) "< *prtdir* "> ) THEN
                                                                    09819
         errors _ gtjoosf                                           09820
      ELSE errors _ 0;                                              09821
      IF NOT jfn _ lgetjfn (0, &ofilnam, $txtext, errors, $lit) THEN
                                                                    09822
         SIGNAL (ofilerr, $lit);                                    09823
      CASE device OF                                                09824
         = optydv, = opxtdv, = oprmdv:                              09825
            oqapfg _  FALSE;                                        09826
         ENDCASE                                                    09827
            IF oqapfg THEN BEGIN                                    09828
            !gtfdb( jfn, 1B6+1, $r3);                               09829
            oqapfg _ IF r3.fdbnxf THEN FALSE ELSE TRUE;             09830
            END;                                                    09831
      gproc _ 0;                                                    09832
      CASE device OF                                                09833
         = opcmdv:                                                  09834
            BEGIN                                                   09835
            opentype _ comtyp;                                      09836
            % Get address of GCOUT (diagram producing procedure); if
            it isn't there, we must load the graphics system for
            diagram formatting.  %                                  09837
               IF NOT ddtlookup($"GCOUT", FALSE, % get procedure in
               user program. %: gproc) THEN                         09838
                  BEGIN                                             09839
                  % Must load graphics program %                    09840
                  % Increase buffer size to 30 pages. %             09841
                     IF NOT gpbsz( 30 ) THEN                        09842
```

```
                        err($"Cannot set user program buffer for
                        graphics");                                    09843
                    % Reset the user program buffer. %                 09844
                        gpgmrst();                                     09845
                        dismes(2,$"all user programs deleted except
                        graphics");                                    09846
                    % Load the user program with GRAPHICS code. %      09847
                        *gname* _ "programs, graphics";                09848
                        ON SIGNAL ELSE                                 09849
                            BEGIN                                      09850
                            ON SIGNAL ELSE                             09851
                            err($"Cannot load GRAPHICS programs");      09852
                            END;                                       09853
                        getuprog($gname);                              09854
                        ON SIGNAL ELSE;                                09855
                    % Get address of GCOUT (diagram producing
                    procedure) %                                       09856
                        IF NOT ddtlookup($"GCOUT", FALSE, % get
                        procedure in user program. %: gproc) THEN
                        err($"Cannot find gcout address");             09857
                    END;                                               09858
                END;                                                   09859
            = optydv, = opxtdv: opentype _  lpttype;                   09860
            =oprmdv: opentype _ netype;                                09861
            ENDCASE opentype _ chrtyp;                                 09862
        IF NOT sysopen (jfn, IF oqapfg THEN append ELSE write,
        opentype, $lit) THEN                                           09863
            BEGIN                                                      09864
            reljfn (jfn);                                              09865
            SIGNAL (ofilerr, $lit);                                    09866
            END;                                                       09867
    opinit (&da, jfn, device, opflags, gproc, comexflag);              09868
    tp[0] _ da.dacsp; tp[1] _ 1;                                       09869
    IF NOT (prcjfn _ sgtjfn(gtjprf, IF exp THEN $xopname ELSE $opname,
    $lit)) THEN                                                        09870
        SIGNAL(prcerr, $lit);                                          09871
    errors _ processor                                                 09872
        (0, $tp, &da, odtyp, jfn, prcjfn);                             09873
    % close output file %                                             09874
        IF NOT sysclose (jfn, $lit) THEN                               09875
            BEGIN                                                      09876
            reljfn (jfn);                                              09877
            dismes (2, $lit);                                          09878
            GOTO STATE;                                                09879
            END;                                                       09880
    % if waiting at page breaks, wait before returning %               09881
        IF opflags.opwtpb THEN DO                                      09882
            jfn _ input()                                              09883
        UNTIL jfn # LF;                                                09884
    RETURN;                                                            09885
    END.   %%
```

```
                                                                    09886
FINISH                                                              01512
%old%                                                               08206
   (xassim) PROCEDURE                                               01513
      (target, src1, src2, levstg, vspec1, vspec2, usqcod, cacode);
                                                                    01514
      LOCAL newtgt, newsrc, lstlev, toplev, sqgwrk;                 01515
      REF levstg, sqgwrk;                                           01516
      RETURN;  %******** RETURN FOR NOW ********%                   01517
      &sqgwrk _ openseq (src1, src2, vspec1, vspec2, usqcod, cacode);
                                                                    01518
      IF (newsrc _ seqgen(&sqgwrk)) # endfil THEN                   01519
         BEGIN                                                      01520
         newtgt _                                                   01521
            IF newsrc.stastr THEN                                   01522
               instat (target, newsrc.stadr, &levstg)               01523
            ELSE copstat(target, newsrc, &levstg);                  01524
         toplev _ lstlev _                                          01525
            IF sqgwrk.swclvl = 0 THEN 1 ELSE sqgwrk.swclvl;         01526
         WHILE (newsrc _ seqgen(&sqgwrk)) # endfil DO               01527
            BEGIN                                                   01528
            *levstg* _ NULL;                                        01529
            CASE sqgwrk.swclvl OF                                   01530
               =lstlev: NULL;                                       01531
               >lstlev:                                             01532
                  BEGIN                                             01533
                  *levstg* _ 'D;                                    01534
                  lstlev _ lstlev + 1;                              01535
                  END;                                              01536
               <lstlev:                                             01537
                  WHILE lstlev > toplev DO                          01538
                     BEGIN                                          01539
                     *levstg* _ *levstg*, 'U;                       01540
                     IF ((lstlev _ lstlev - 1) = sqgwrk.swclvl) THEN
                                                                    01541
                        EXIT LOOP 1;                                01542
                     END;                                           01543
               ENDCASE NULL;                                        01544
            newtgt _                                                01545
               IF newsrc.stastr THEN                                01546
                  instat (newtgt, newsrc.stadr, &levstg)            01547
               ELSE copstat (newtgt, newsrc, &levstg);              01548
            END;                                                    01549
         END;                                                       01550
      closeseq (&sqgwrk);                                           01551
      RETURN;                                                       01552
      END.   %%
```

01553